Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cosrev

Survey

DAG-based attack and defense modeling: Don't miss the forest for the attack trees

Barbara Kordy[a,b,*], *Ludovic Piètre-Cambacédès*[c], *Patrick Schweitzer*[a]

[a] *University of Luxembourg, SnT, 6, rue Coudenhove-Kalergi, 1359, Luxembourg*
[b] *IRISA, INSA Rennes, Campus Beaulieu, 35042 Rennes, France*
[c] *EDF France, Research and Development Department, 1, avenue Général de Gaulle, 92141 Clamart, France*

HIGHLIGHTS

- We present an overview of attack and defense modeling techniques based on DAGs.
- We summarize existing methodologies and compare their features.
- We propose a taxonomy of the described formalisms.
- We support the selection of a modeling technique depending on user requirements.
- We point out future research directions in the field of graphical security modeling.

ARTICLE INFO

Article history:
Received 23 May 2013
Received in revised form
18 July 2014
Accepted 20 July 2014
Published online 11 August 2014

Keywords:
Graphical models for security
Attack trees
Bayesian networks
Attack and defense modeling
Quantitative and qualitative security assessment
Security measures

ABSTRACT

This paper presents the current state of the art on *attack and defense modeling approaches that are based on directed acyclic graphs (DAGs)*. DAGs allow for a hierarchical decomposition of complex scenarios into simple, easily understandable and quantifiable actions. Methods based on threat trees and Bayesian networks are two well-known approaches to security modeling. However there exist more than 30 DAG-based methodologies, each having different features and goals.

The objective of this survey is to summarize the existing methodologies, compare their features, and propose a taxonomy of the described formalisms. This article also supports the selection of an adequate modeling technique depending on user requirements.

Contents

* *Corresponding author at:* University of Luxembourg, SnT 6, rue Coudenhove-Kalergi, 1359, Luxembourg. Tel.: +352 4666445506.
E-mail addresses: basia.kordy@gmail.com (B. Kordy), ludovic.pietre-cambacedes@edf.fr (L. Piètre-Cambacédès), patrick.schweitzer@uni.lu (P. Schweitzer).

http://dx.doi.org/10.1016/j.cosrev.2014.07.001
1574-0137/

1. Introduction

Graphical security models provide a useful method to represent and analyze security scenarios that examine vulnerabilities of systems and organizations. The great advantage of graph-based approaches lies in combining user friendly, intuitive, visual features with formal semantics and algorithms that allow for qualitative and quantitative analyses. Over the course of the last two decades, graphical modeling has attracted the attention of numerous security and formal methods experts. It has quickly become a stand-alone research area with dedicated dissemination events [1] as well as related national and international research projects [2–9]. Graphical models constitute a valuable support tool to facilitate threat assessment and risk management of real-life systems. Thus, they have also become popular in the industrial

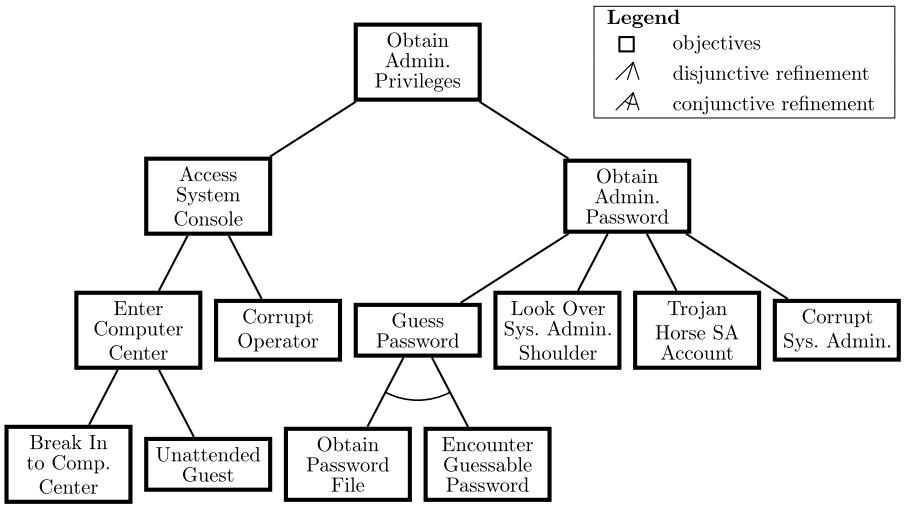

Fig. 1 – A threat logic tree taken from [23]: Obtaining administrator privileges on a UNIX system.

sector. Notable application domains of graphical models include security analysis of supervisory control and data acquisition (SCADA) systems [10–12], voting systems [13,14], vehicular communication systems [15,16], Internet related attacks [17,18], secure software engineering [19], and socio-technical attacks [20–22].

In this paper we focus on graphical methods for the analysis of attack and defense scenarios. We understand attack and defense scenarios in a general sense: they encompass any malicious action of an attacker who wants to harm or damage another party or its assets as well as any defense or countermeasure that could be used to prevent or mitigate such malicious actions. In 1991, Weiss [23] introduced threat logic trees as the first graphical attack modeling technique. The obvious similarity of threat logic trees to fault trees [24] suggests that graph-based security modeling has its roots in safety modeling. Weiss' approach can be seen as the origin of numerous subsequent models, including attack trees [25,26] which are nowadays one of the most popular graphical security models.

Today, more than 30 different approaches for the analysis of attack and defense scenarios exist. Most of them extend the original model of threat logic trees in one or several dimensions which include defensive components, timed and ordered actions, dynamic aspects, and different types of quantification. Moreover, methods for computation of various security related parameters, such as the cost, the impact or likelihood of an attack, the efficiency of necessary protection measures, or the environmental damage of an attack, have been developed or adapted.

This survey concentrates on formalisms based on directed acyclic graphs (DAGs), rather than on arbitrary graphs. Described approaches can be divided into two main classes: formalisms derived from or extending threat trees, and formalisms based on Bayesian networks. The model creation in all threat tree-based methodologies starts with the identification of a feared event represented as the root node. Then, the event's causes or consequences, depending on the specific approach, are deduced and depicted as refining nodes. The refinement process is illustrated in Fig. 1, which recreates the first threat tree model proposed by Weiss [23]. The DAG structure allows to use refinements with a customizable level of detail. The root of a DAG is refined as long as the refining children provide useful and adequate information about the modeled scenario. Refinements paired with the acyclic structure allow for modularization which in turn allows different experts to work in parallel on the same model. This is highly appreciated in case of large-scale, complex models, where analysis of different parts requires different types of expertise. A big advantage of the DAG-based approaches is that they are fairly scalable. They do not suffer from the state space explosion problem, which is common for models based on general graphs with cycles. In the case of trees, most of the analysis algorithms are linear with respect to the number of nodes of the model. Due to multiple incoming edges, this property is no longer true for DAGs and the complexity of analysis methods might, in theory, be exponential. However in practice, this is still acceptable, since the exponents can be kept small due to the underlying cycle-free structure. This is, for instance, the case for Bayesian inference algorithms used for the analysis of security models based on Bayesian networks. Fig. 2 depicts a simple Bayesian attack graph borrowed from [27] and illustrates how to compute the unconditional probability of a vulnerability exploitation.

This paper surveys DAG-based graphical formalisms for attack and defense modeling. These formalisms provide a systematic, intuitive, and practical representation of a large amount of possible attacks, vulnerabilities and countermeasures, while at the same time allowing for an efficient formal and quantitative analysis of security scenarios. The contribution of this work is to provide a complete overview of the field and systematize existing knowledge. More specifically, the survey

- presents the state of the art in the field of DAG-based graphical attack and defense modeling;
- identifies relevant key aspects allowing to compare different formalisms;
- proposes a taxonomy of the presented approaches, which helps in selecting an appropriate formalism;

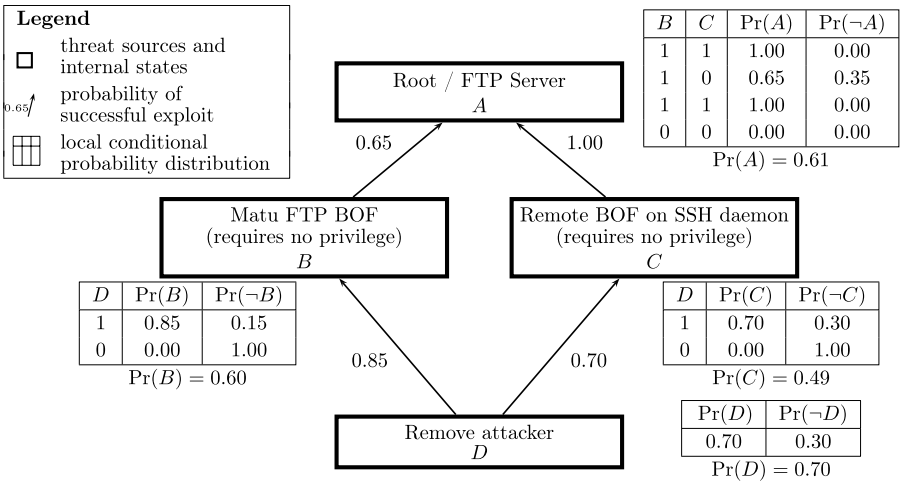

Fig. 2 – Bayesian attack graph taken from [27]: A test network with local conditional probability distributions (tables) and updated unconditional probabilities (below each table).

- lays a foundation for future research in the field, with the goal to prevent reinvention of already existing features.

In Section 2, we introduce terminology used in the field of graph-based security modeling and provide a template for the description of the formalisms. Section 3 is the main part of the survey and presents existing DAG-based attack and/or defense modeling approaches. In Section 4, we provide a concise tabular overview of the presented formalisms. We illustrate how to use the tables in order to select the most relevant modeling technique, depending on the application requirements. Section 5 briefly mentions alternative graphical security models. We close the survey with concluding section, which summarizes our findings and proposes future research directions in the field.

2. Preliminaries

In this section we introduce our terminology and make a link to existing definitions and concepts. We then present and define the aspects on the basis of which we have analyzed the different formalisms. We conclude with a detailed description of how formalisms from Section 3 are described.

2.1. Keywords and terminology

When examining different models in the same context, it is imperative to have a common language. Over the last 20 years, numerous concepts and definitions have emerged in the field of graphical security modeling. This section is intended to introduce the language used in this paper, and to serve as a quick reference guide over the most commonly occurring concepts. Our goal here is not to point out the differences in definitions or other intricate details.

Attack and defense modeling. By techniques for attack and defense modeling we understand formalisms that serve for representation and analysis of malicious behavior of an *attacker* and allow to reason about possible defending strategies of the attacker's opponent, called the *defender*. In our survey we use attacks in a very broad sense. Attacks can also be thought of as *threats*, *obstacles*, and *vulnerabilities*. On the contrary, defenses can appear in the form of *protections*, *mitigations*, *responses*, and *countermeasures*. They oppose, mitigate or prevent attacks.

Nodes. *Nodes*, also called *vertices*, are one of the main components of graph-based security models. They are used to depict the concept that is being modeled. Nodes may represent *events*, *goals*, *objectives*, and *actions*. Depending on whether the models are constructed in an inductive or deductive way, nodes may also express *causes* or *consequences*.

Root node. In a rooted DAG (and therefore in any tree) the *root* is the single designated node that does not have any predecessor. From it all other nodes can be reached via a directed path. This distinguished node usually depicts the entire concept which is being modeled. In the context of security models, various existing names for this special node include *top event*, *main goal*, *main consequence*, *main objective* or *main action*.

Leaf nodes. In a DAG, nodes that do not have any children are called *leaves*. They usually display an atomic component of a scenario that is no longer refined. They are also called *primary events*, *basic components*, *elementary attacks*, *elementary components* or *basic actions*.

Edges. Edges are the second main component of graph-based security models. They link nodes with each other and, in this way, determine relations between the modeled concepts. Edges are also called *arcs*, *arrows*, or *lines*. In some models, edges may have special semantics and may detail a cause–consequence relation, a specialization or some other information.

Connectors. Connectors usually specify more precisely how a parent node is connected with its children. A connector might be a set of edges or a node of a special type. Connectors are also called *refinements* or *gates*. Some examples include: AND, OR, XOR, *k*-out-of-*n*, priority AND, triggers, etc.

Priority AND. A *priority AND* (PAND) is a special kind of AND connector which prescribes an order in which the nodes are to be treated. The origin of the prescribed order is usually

time or some priority criterion. The PAND is also called an *ordered-AND*, an *O-AND* or a *sequential AND*. Sometimes the underlying reason behind the priority is specified as in the case of the *time-based AND*.

Attributes. *Attributes* represent aspects or properties that are relevant for quantitative analysis of security models. Examples of attributes, sometimes also called *metrics*, include: impact of an attack, costs of necessary defenses, risk associated with an attack etc. Proposed computation methods range from versatile approaches that can be applied for evaluation of a wide class of attributes, to specific algorithms developed for particular measures. An example of the former is the formalization of an attribute domain proposed in [28], which is well suited for calculation of any attribute whose underlying algebraic structure is a semi-ring. An example of the latter are the specific methods for probability computation proposed in [29].

2.2. Examined aspects

One of the goals of this paper is to provide a classification of existing formalisms for attack and defense modeling. Thus, all approaches described in Section 3 were analyzed based on the same 13 criteria, which we refer to as *aspects* and define in this section.

The formalisms are grouped according to the following two main aspects:

1. **Attack and/or defense modeling:** *Attack* modeling techniques are focused on an attacker's actions and vulnerabilities of systems; *defense* modeling techniques concentrate on defensive aspects, such as detection, reaction, responses, and prevention.
2. **Static or sequential approaches:** *Sequential* formalisms take temporal aspects, such as dynamics time variations, and dependencies between considered actions, such as order or priority, into account; *static* approaches cannot model any of such relations.

The above two aspects provide a partition of all considered approaches. Furthermore, they correspond to questions that a user selecting a suitable formalism is most likely to ask, namely *'What do we want to model?'* and *'How do we want to model?'*. The proposed classification allows a reader to easily make a primary selection and identify which formalisms best fit his needs.

Besides the two main aspects, each formalism is analyzed according to additional criteria, listed in Table 1. All aspects taken into account in our work, can be grouped into three categories:

- Aspects relating to the formalism's *modeling capabilities*, i.e., what we can model: attack or defense modeling, sequential or static modeling, quantification, main purpose, extensions.
- Aspects relating to the formalism's *characteristics*, i.e., how we can model: structure, connectors, formalization.
- Aspects related to the formalism's *maturity and usability*: tool availability, case study, external use, paper count, year.

In Table 1, we define all 13 aspects in the form of questions and provide possible values that answer the questions.

2.3. Template of the formalism descriptions

The description of each formalism presented in Section 3 complies with the following template.

General presentation. The first paragraph mentions the name of the formalism, its authors, as well as it lists main related papers. The year when the approach was proposed is given. Here we also present the main purpose for which the technique was introduced. If nothing is indicated about the formalism structure, it means that it is a generic DAG. If the structure is more specifically a tree, then it is indicated either in the formalism's name or in the first paragraph of the description.

Main features. In the second paragraph, we briefly explain the main features of the formalism, in particular what its added features are with respect to the state of the art at the time of its invention. Moreover, we state whether the modeling technique is formalized, i.e., whether it complies with proper mathematical definitions.

Quantification. Next, we focus on quantitative aspects of the considered methodology. We explain whether the formalism is tailored for a couple of specific parameters or metrics, or whether a general framework has been introduced to deal with computations. In the first case, we list relevant attributes, in the second case, we briefly explain the new algorithms or calculation procedures.

Practical aspects. When relevant, we mention industrialized or prototype software tools supporting the described approach. We also indicate when real or realistic scenarios have been modeled and analyzed with the help of the described approach. In this paragraph, we also refer to large research projects and Ph.D. theses applying the methodology. This paragraph is optional.

Additional remarks. We finish the formalism description by relating it to follow-up methodologies. We point out the formalism's limitations that have been identified by its authors or other researchers from the field. In this part we also point out various other peculiarities related to the formalism. This paragraph is optional.

3. Description of the formalisms

This section constitutes the main part of this survey. It describes numerous DAG-based approaches for graphical attack and defense modeling according to the template outlined in Section 2.3. Models gathered within each subsection are ordered chronologically, with respect to the year of their introduction.

3.1. Static modeling of attacks

3.1.1. Attack trees

Inspired by research in the reliability area, Weiss [23] in 1991 and Amoroso [30] in 1994 proposed to adopt a tree-based concept of visual system reliability engineering to security. Today, *threat trees* [30–34], *threat logic trees* [23], *cyber threat trees* [35], *fault trees* for attack modeling [36], and the *attack specification language* [17] can be subsumed under *attack trees*,

Table 1 – Table summarizing aspects taken into account in formalism description.

Aspect	Aspect description	Possible values	Value explanation
	Is the formalism offensively or defensively oriented?	Attack	Only attack modeling
		Defense	Only defense modeling
		Both	Integrates attack and defense modeling
Static or sequential	Can the formalism deal with dependencies and time varying scenarios?	Static	Does not support any dependencies
		Sequential	Supports time and order dependencies
Quantification	Can numerical values be computed using the formalism?	Versatile	Supports numerous generic and diverse metrics
		Specific	Dedicated, tailored for (a couple of) specific metrics
		No	Does not support quantification
Main purpose	Why was the formalism invented?	Sec. mod.	General security modeling
		Unification	Unification of existing formalisms
		Quantitative	Provide better methods for quantitative analysis
		Risk	Support risk assessment
		Soft. dev.	Support secure software development
		Int. det.	Automated intrusion detection and response analysis
		Req. eng.	Support security requirements engineering
Extensions	What are the added features of the formalism with respect to the state of the art?	Structural	New connectors, extended graph structure
		Computational	How the formalism handles computations (e.g., top down)
		Quantitative	Which computations can be performed (e.g., specific attributes)
		Time	The formalism can handle time dependencies
		Order	The formalism can handle order dependencies
		New formalism	Entirely new formalism
Structure	Which graphical structure is the formalism based on?	Tree	Tree (possibly with repeated nodes)
		DAG	Directed acyclic graph
		Unspecified	It is not specified whether the models are DAGs or trees
Connectors	What type of connectors does the formalism use?	List of connectors	AND, OR, trigger, sequential AND, ordered-AND, priority AND, k-out-of-n, OWA nodes, split gate, countermeasures, counter leaves, dependence edges
Formalization	Is the formalism formally defined?	Formal	Defined using a mathematical framework; with clear syntax and semantics
		Semi-formal	Parts of the definitions are given verbally, parts are precise
		Informal	Models only verbally described
Tool availability	Does a software tool supporting the formalism exist?	Commercial	A commercial software tool exists
		Prototype	A prototype tool exists
		No	No implementation exists
Case study	Do papers or reports describing case studies exist?	Real(istic)	Real or realistic case study has been documented
		Toy case study	Toy case study has been described
		No	No documented case study exist
External use	Do papers or reports having a disjoint set of authors from the formalism inventors exist?	Independent	People and institutions who did not invent the formalism have used it
		Collaboration	The formalism has been used by external researchers and institutions in collaboration with its inventors
		No	The formalism has only been used by its inventors or within the institution where it was invented
Paper count	How many papers on the formalism exist?	Number	Number of papers that have been identified[1]
Year	In which year was the formalism first published?	Year	Before 2013

[1]Different versions of the same paper (e.g., an official publication and a corresponding technical report) have been counted as the same publication.

which are AND–OR tree structures used in graphical security modeling. The name attack trees was first mentioned by Salter et al. in 1998 [25] but is often only attributed to Schneier and cited as [26,37].

In the attack tree formalism, an attacker's main goal (or a main security threat) is specified and depicted as the root of a tree. The goal is then disjunctively or conjunctively refined into sub-goals. The refinement is repeated recursively,

until the reached sub-goals represent basic actions. Basic actions correspond to atomic components, which can easily be understood and quantified. Disjunctive refinements represent different alternative ways of how a goal can be achieved, whereas conjunctive refinements depict different steps an attacker needs to take in order to achieve a goal [38]. In 2005, Mauw and Oostdijk formalize attack trees by defining their semantics and specifying tree transformations consistent with their framework [28]. Kienzle and Wulf present an extensive general procedure for tree construction [39] while other researchers are engaged in describing how to generate attack tree templates using *attack patterns* [40,41]. Most recently, the problem of automated generation of attack trees has started to attract the attention of scientific as well as industrial communities [42,43].

Quantification of security with the help of attack trees is a very active topic of research [44]. A first simple procedure for quantification using attack trees was proposed by Weiss [23] and is based on a bottom-up algorithm. In this algorithm, values are provided for all leaf nodes and the tree is traversed from the leaves towards the root in order to compute values of the refined nodes. Depending on the type of refinement, different functional operators are used to combine the values of the children. This procedure allows to analyze simple aspects, such as the costs of an attack, the time of an attack or the necessary skill level [23,30,25,26,10,45,46,28,47,29,48,49,15,50–52,12,44]. Whenever more complicated attributes, such as probability of occurrence, probability of success, risk or similar measures are analyzed, additional assumptions, for example mutual independence of all leaf nodes, are necessary, or methods different from the bottom-up procedure have to be used [26,10,53,48,29,54,15,50,55,51,56,35,57–62]. Propagation of fuzzy numbers that model fuzzy preference relations has initially been proposed in [63] and extended in [64]. Using Choquet integrals it is possible to take interactions between nodes into account.

Commercial software for attack tree modeling, such as SecurITree [65] from Amenaza or AttackTree+ [66] from Isograph provides a large database of attack tree templates. Academic tools, including SeaMonster [67] developed within the SHIELDS project [2] offer visualization and library support. Attack trees may occur in the Security Quality Requirements Engineering (SQUARE) methodology [68]. The entire methodology and therefore visualization of attack trees are supported by the SQUARE tool [69]. AttackDog [70] was developed as a prototype software tool for managing and evaluating attack trees with voting systems in mind but is believed to be much more widely applicable to evaluating security risks in systems [71]. Numerous case studies [40,17,10,72,73,45,74,46,68,16,75,11,76–80,22,49,15,18,81–86,21,13,87–90,62] account for the applicability of the attack tree methodology. Attack trees are used in large international research projects [91,2,3]. They have been focus of various Ph.D. and Master theses [92–114]. Attack tree modeling goes beyond the academic world and is finding its way in industrial practices, especially those related to critical sectors [115,116].

Since attack trees only focus on static modeling and only take an attacker's behavior into account, numerous extensions that include dynamic modeling and a defender's behavior exist. Except for formalisms involving Bayesian inference techniques, all other DAG-based formalisms refer back to the attack tree methodology. They point out a need for modeling defenses, dynamics, and ordered actions, as well as propose computation procedures for probability or highly specified key figures. Neither the name attack trees, nor the initial formalization of Mauw and Oostdijk is universally accepted. Some researchers consider attack trees, threat trees or fault trees to essentially be the same [117–121,36] while other researchers point out specific differences [50,122]. As common ground all mentioned methodologies use an AND–OR tree structure but are divided on what the tree can actually model (attacks, vulnerabilities, threats, failures, etc.).

3.1.2. *Augmented vulnerability trees*

Vulnerability trees [123] have been proposed by Vidalis and Jones in 2003 to support the decision making process in threat assessment. Vulnerability trees are meant to represent hierarchical interdependence between different vulnerabilities of a system. In 2008, Patel, Graham, and Ralston [124] extended this model to *augmented vulnerability trees* which combine the concepts of vulnerability trees, fault tree analysis, attack trees, and cause–consequence diagrams. The aim of augmented vulnerability trees is to express the financial risk that computer-based information systems face, in terms of a numeric value, called "degree of security".

The root of a vulnerability tree is an event that represents a vulnerability; the branches correspond to different ways of exploiting it. The leaves of the tree symbolize steps that an attacker may perform in order to get to the parent event. The model, which is not formally defined, uses only AND and OR connectors depicted as logical gates. Vulnerability trees are very similar to attack trees, they differ in how the root event is defined (vulnerability event vs. an attacker's goal). A step-wise methodology consisting of a sequence of six steps is proposed in [124] to create an augmented vulnerability tree and analyze security related indexes.

The authors of [123] propose a number of attributes on vulnerability trees, including: complexity value (the smaller number of steps that an attacker has to employ in order to achieve his goal), educational complexity (qualifications that an attacker has to acquire in order to exploit a given vulnerability), and time necessary to exploit a vulnerability. However, the paper [123] does not detail on how to compute these attributes. In [124], the model is augmented with two indexes: the threat–impact index and the cyber-vulnerability index. The first index, represented by a value from $[0, 100]$, expresses the financial impact of a probable cyber threat. The lower the index, the smaller is the impact from a successful cyber attack. The second index, also expressed by a value from $[0, 100]$, represents system flaws or undesirable events that would help an intruder to launch attacks. The lower this index, the more secure the system is.

In [12], the augmented vulnerability tree approach has been used to evaluate risks posed to a SCADA system exposed to the mobile and the Internet environment.

3.1.3. *Augmented attack trees*

In 2005, Ray and Poolsappasit[1] first developed *augmented attack trees* to provide a probabilistic measure of how far an attacker has progressed towards compromising a system [125].

[1] In early papers spelled Poolsapassit [125,126].

This tree-based approach was taken up by H. Wang et al. in 2006 and extended to allow more flexibility in the probabilistic values provided for the leaf nodes [127]. When again publishing in 2007, Poolsappasit and Ray used a different definition of augmented attack trees to be able to perform a forensic analysis of log files [126]. Using the second definition of augmented attack trees, J. Wang et al. performed an analysis of SQL injection attacks [128] and Distributed Denial of Service (DDoS) attacks [129]. They also extended augmented attack trees further to measure the quality of detectability of an attack [130]. The authors of [131] and [132] formalized attack trees as AND–OR structure where every node is interpreted to answer a specific binary question. This formalization is then again extended to augmented attack trees by adding to every node an indicator variable and an additional value with the help of which the residual damage is computed. On the enhanced structure they are able to optimize how to efficiently trade-off between spent money and residual damage.

The various ways of defining augmented attack trees are based on attack trees (Section 3.1.1). In the first definition, attack trees are augmented by node labels that quantify the number of compromised subgoals on the most advanced attack path as well as the least-effort needed to compromise the subgoal on the most advanced path to be able to compute the probability of attack [125]. H. Wang et al. generalized this definition from integer values to general weights. Both approaches include tree pruning and tree trimming algorithms to eliminate irrelevant nodes with respect to intended operations (behavior) of a user [127]. In the second definition, attack trees are augmented by descriptive edge labels and attack signatures. Each edge defines an atomic attack which is described by the label and represents a state transition from a child node to the corresponding parent. An attack signature is a sequence of groups of incidents, from which a sequence of incidents can be formed, which constitutes an atomic attack. The sequences are then exploited to filter log files for relevant intrusion incidences [126] and used to describe state transitions in SQL injection attacks using regular expressions [128]. Moreover they are exploited to model state transition in DDoS attacks [129] and adapted to provide a measure for quality of service detection, called quality of detectability [130]. In an extension of the third definition [132] the system administrator's dilemma is thoroughly examined. The purpose of this extension is to be able to compute a bounded minimization of the cost of the security measures while also keeping the residual damage at a minimum.

Augmented attack trees were designed with a specific quantitative purpose in mind. The first formalization of augmented attack trees was introduced to compute the probability of a system being successfully attacked. Additionally to increasing the descriptive capabilities of the methodology, the second definition is accompanied by several algorithms that help compute the quality of detectability in [130]. As mentioned before, the third definition targets solving the system administrator's dilemma. This is achieved by using a simplistic cost model and a multi-objective optimization algorithm which guides the optimization process of which security hardening measures best to employ.

The authors of the first formalism state that attempts by system administrators to protect the system will not change the outcome of their analysis. A similar shortcoming is suggested for the second formalization.

3.1.4. *OWA trees*

In 2005, Yager proposed to extend the AND and OR nodes used in attack trees by replacing them with ordered weighted averaging (OWA) nodes. The resulting formalism is called *OWA trees* [29] and it forms a general methodology for qualitative and quantitative modeling of attacks.

Regular attack trees make use of two (extreme) operators only: AND (to be used when *all* actions need to be fulfilled in order to achieve a given goal) and OR (to be used when the fulfillment of *at least one* action is sufficient to reach a desired result). OWA operators represent quantifiers such as *most*, *some*, *half of*, etc. Thus, OWA trees are well suited to model uncertainty and to reason about situations where the number of actions that need to be satisfied is unknown. OWA trees are static in the sense that they do not take interdependencies between nodes into account. They have been formally defined in [29] using the notion of an *OWA weighting vector*. Since AND and OR nodes can be seen as special cases of OWA nodes, mathematically, attack trees form a subclass of OWA trees. Therefore, algorithms proposed for OWA trees are also suitable for the analysis of attack trees.

In [29], Yager provides sound techniques for the evaluation of success probability and cost attributes on OWA trees. For the probability attribute, he identifies two approaches that can be explained using two different types of attackers. The first approach assumes that the attacker is able to try all available actions until he finds one that succeeds. Since in most situations such an assumption is unrealistic, the author proposes a second model, where an attacker simply chooses the action with the highest probability of success. Furthermore, [29] presents two algorithms for computing the success probability attribute: one assumes independent actions which leads to a simpler calculation procedure, the other can deal with dependent actions. Finally, the author discusses how to join the two attributes together, in order to correctly compute the cheapest and most probable attack.

In [63], Bortot, Fedrizzi, and Giove proposed the use of Choquet integrals in order to reason about OWA trees involving dependent actions.

3.1.5. *Parallel model for multi-parameter attack trees*

In 2006, Buldas, Laud, Priisalu, Saarepera, and Willemson initiated a series of papers on rational choice of economically relevant security measures using attack trees. The proposed model is called *multi-parameter attack trees* and was first introduced in [53]. Between 2006 and 2013, researchers from different research institutes in Estonia published seven follow-up papers [14,133,54,134–137], extending and improving the original model proposed in [53].

Most approaches for quantitative analysis using attack trees, prior to [53], focus on one specific attribute, e.g., cost or feasibility of an attack. In reality, interactions between different parameters play an important role. The aim of the mentioned series of papers was to study how tree computations must be done when several interdependent parameters are considered. The model of multi-parameter attack trees assumes that the attacker behavior is rational. This means that attacks are considered unlikely if their costs are greater than the related benefits and that the attacker always chooses the most profitable way of attacking. The parallel model for

multi-parameter attack trees has been studied in [53,14,133, 54,135,104]. This model assumes that all elementary attacks take place simultaneously, thus the attacker does not base his decisions on success or failure of some of the elementary attacks.

Multi-parameter attack trees concentrate on the attribute called expected attacker's outcome. This outcome represents a monetary gain of the attacker and depends on the following parameters: gains of the attacker in case the attack succeeds, costs of the attack, success probability of the attack, probability of getting caught and expected penalties in case of being caught. First, a game theoretical model for estimation of the expected attacker's outcome was proposed by Buldas et al. [53], where values of all parameters are considered to be precise point estimates. In [133], Jürgenson and Willemson extend the computation methods proposed in [53] to the case of interval estimations. Later it turned out that the computational model from [53] was imprecise and inconsistent with the mathematical foundations of attack trees introduced in [28]. Hence, an improved approach for the parallel attack tree model was proposed by Jürgenson and Willemson [54]. Since this new approach requires exponential running time to determine possible expected outcome of the attacker, an optimization solution, based on a genetic algorithm for fast approximate computations, has been proposed by the same authors in [135].

In [14], Buldas, and Mägi applied the approach developed in [53] to evaluate the security of two real e-voting schemes: the Estonian E-voting System in use at the time (EstEVS) and the Secure Electronic Registration and Voting Experiment (SERVE) performed in the USA in 2004. A detailed description of this case study is given in the Master thesis of Mägi [102]. A prototype computer tool supporting the security analysis using the multi-parameter attack trees has been implemented [138] and described in [139].

In Section 3.2.9, we describe the serial model for multi-parameter attack trees, which extends the parallel model with an order on the set of elementary components.

3.1.6. *Extended fault trees*

Extended fault trees (EFTs) were presented by Masera et al. at the ESREL conference in 2007 [140] and published in an extended version as a journal paper [141] issued in 2009. The formalism aims at combining malicious deliberate acts, which are generally captured by attack trees (Section 3.1.1), and random failures, which are often associated with classical fault trees (Section 3.1.1).

Extended fault trees and attack trees are structurally similar. The main difference between the two formalisms is in the type of basic events that can be modeled. In EFT basic events can represent both non-malicious, accidental failures as well as attack steps or security events. Basic events of attack trees usually correspond to malicious attacker's actions only. Logical AND and OR gates are explicitly represented in the same way as in classical fault trees. A step-by-step model construction process is described in [141], defining how existing fault-trees can be extended with attack-related components to form extended fault tree models. The modeling technique complies with proper mathematical foundations, directly issued from fault trees as defined in the safety and reliability area.

Quantification capabilities are focused on the computation of the probability of occurrence of the top-event (root node). Generic formulas from fault tree quantitative analysis are recalled in [141], including treatment of independent or mutually exclusive events. However, no concrete examples of quantification are provided.

A simple example, analyzing the different failure and attack scenarios leading to the release of a toxic substance by a chemical plant, is described in [141]. No particular tool has been developed to support extended fault trees, however, all classical fault tree tools may be used directly.

One of the limitations explicitly stressed by the inventors of extended fault trees is that they do not take into account time dynamics.

3.2. *Sequential modeling of attacks*

3.2.1. *Cryptographic DAGs*

Meadows described *cryptographic DAGs* in 1996 (proceedings published in 1998), in order to provide a simple representation of an attack process [142]. The purpose of the formalism is limited to visual description. The attack stages of the overall attack process correspond to the nodes of a DAG. The difficulty of each stage is shown by a color code. In 1996, the novelty of cryptographic DAGs was to provide a simple representation technique of sequences and dependencies of attack steps towards a given attacker's objective.

From a modeling point of view, each stage (represented as a colored box) contains a textual description of atomic actions needed for the realization of the stage. Arrows represent dependencies between the boxes. A simple arrow indicates that one stage is needed to realize another stage. Two arrows fanned out symbolize that one stage enables another one repeatedly. More generally speaking, cryptographic DAGs are an informal formalism targeted at high level system descriptions.

Cryptographic DAGs do not support any type of quantification.

Cryptographic DAGs have been used in [142] to demonstrate attacks on cryptographic protocols (with SSL and Needham–Schroeder scheme as use cases), however this representation technique may be used to model other types of attacks as well.

This formalism allows the representation of sequences of attack steps, and dependencies between those steps, but cannot capture static relations like AND and OR. Moreover, the clarity and usability of the models depend heavily on the text inside the boxes, which is not standardized.

3.2.2. *Fault trees for security*

Fault tree analysis was born in 1961 and has initially been developed into a safety, reliability, and risk assessment methodology [143,24,144,145]. A short history of non-security related fault trees was published by Ericson II [146] in 1999. Fault trees have also been used for software analysis [147–150] and were even equated with attack trees by Steffen and Schumacher [36]. In 2003, however, Brooke, and Paige adopted *fault*

trees for security, extending the classical AND–OR structure of attack trees (Section 3.1.1), to include well-known concepts from safety analysis [151].

Based on an AND–OR structure, three additional connectors (priority AND, exclusive OR and inhibit), specific node types (basic, conditioning, undeveloped, external, and intermediate), as well as transfer symbols (transfer in, transfer out) to break up larger trees are adopted from fault tree analysis in its widest sense. Fault trees for security are an aid to the analysis of security-critical systems, where first an undesired (root) event is identified. Then, new events are constructed by inserting connectors that explicitly identify the relationship of the events to each other. Several rules, like the "no miracle" rule, the "complete the gate" rule, and the "no gate to gate" rule are adopted directly from fault trees. Construction stops when there are no more uncompleted intermediate events. In the end, a completed fault tree serves as an "attack handbook" by providing information about the interactions by which a security critical system fails.

In [151], Brooke, and Paige state that in computer security "it is difficult to assign useful probabilities to the events". Consequently probabilistic quantitative analysis is debatable. Instead the authors recommend to perform risk analysis which answers how the system fails based on the primary events (leaf nodes).

While [151] only provides a toy example, the authors state that any tool used in fault tree analysis can be used. They refer to [152] as a good overview of available programs.

3.2.3. *Bayesian networks for security*

Starting in 2004, different researchers proposed, seemingly independently, to adopt *Bayesian networks*, whose origin lies in artificial intelligence, as a security modeling technique [153–156]. Bayesian networks are also known as *belief network* or *causal network*. In Bayesian networks, nodes represent events or objects and are associated with probabilistic variables. Directed edges represent causal dependencies between nodes. Mathematical algorithms developed for Bayesian networks are suited to solve probabilistic questions on DAG structures. They are aimed at keeping the exponent small when the computing algorithm is exponential and reduce to polynomial algorithms if the DAG is actually a tree.

According to Qin and Lee, the objective of Bayesian networks for security is to "use probabilistic inference techniques to evaluate the likelihood of attack goals and predict potential upcoming attacks" [38]. They proposed the following procedure that converts an attack tree into a Bayesian network. Every node in the attack tree is also present in the Bayesian network. An OR relationship from an attack tree is modeled in the Bayesian network with edges pointing from refining nodes that represent causes into the corresponding refined nodes that represent consequences. Deviating from regular attack trees, an AND relationship is assumed to have an explicit (or implicit) order in which the actions have to be executed. The AND relationship can thus be modeled by a directed path, which starts from the first (according to the order) child and ends with the parent node. Dantu et al. follow a different strategy when using Bayesian networks to model security risk management starting from behavior-based attack graphs[2] [157–160]. When processing multi-parameter attack trees with estimated parameter values (Section 3.1.5) Jürgenson and Willemson use Qin and Lee's conversion of an attack tree to a Bayesian network [133]. An et al. propose to add a temporal dimension and to use *dynamic Bayesian networks* for intrusion detection without specifying how the graph is set up [161]. Althebyan and Panda use knowledge graphs and dependency graphs as basis for the construction of a Bayesian network [162]. They analyze a specific type of insider attack and state that their computational procedures were inspired by Dantu et al. Another approach involving Bayesian networks is described by Xie et al. who analyze intrusion detection systems [163]. They state that the key to using Bayesian networks is to "correctly identify and represent relevant uncertainties" which governs their setup of the Bayesian network.

Bayesian networks are used to analyze security under uncertainty. The DAG structure is of great value because it allows to use efficient algorithms. On the one hand there exist efficient inference algorithms that compute a single query (variable elimination, bucket elimination and importance, which are actually equivalent according to Pouly and Kohlas [164]) and on the other hand there are inference algorithms that compute multiple queries at once (bucket tree algorithm and Lauritzen–Spiegelhalter algorithm). In fact, the efficiency of these algorithms can be seen as main reason to the success of Bayesian networks, since querying general graphs is an NP-hard problem [165,166]. Another strength of Bayesian networks is their ability to update the model, i.e., compute a posteriori distribution, when new information is available.

We have not found any dedicated tools for the analysis of Bayesian networks for security. However, numerous tools exist that allow a visual treatment of standard Bayesian networks. One such tool is the Graphical Network Interface (GeNIE) that uses the Structural Modeling, Inference, and Learning Engine (SMILE) [167]. It was, for example, used in [168] to analyze the interoperability of a very small cluster of services and mentioned as hypothetical use in [169]. Another one, called MulVAL [170], was actually developed for attack graphs (Section 5.2), but used in [163] to implement a Bayesian network model. A third tool, tailored to statistical learning with Bayesian networks is bnlearn [171].

There also exist isolated papers that promote the use of Bayesian networks in security without any relation to attack trees or attack graphs. Houmb et al. quantify security risk level from Common Vulnerability Scoring System (CVSS) estimates of frequency and impact using Bayesian networks [172]. Feng and Xie also use Bayesian networks and provide an algorithm of how to merge two sources of information, expert knowledge, and information stored in databases, into one graph [173]. Note that in this section we have gathered approaches that rely on Bayesian networks whose construction starts from graphs that do not contain any cycles. Graphical models that make use of Bayesian networks and that initially contain cycles are treated in Section 3.2.4, formalisms including defenses are described in Section 3.4.3.

[2] The authors do not appear to make a distinction between attack trees and attack graphs. Since their methodology is only applicable to cycle-free structures and they do not mention how to deal with cycles, we assume that the methodology is actually based on attack DAGs or attack trees.

3.2.4. *Bayesian attack graphs*

Bayesian Attack Graphs combine (general) attack graphs (Section 5.2), with computational procedures of Bayesian networks (Section 3.2.3). However, since Bayesian inference procedures only work on cycle-free structures, the formalism includes instructions on how to remove any occurring cycles. Hence any final Bayesian attack graph is acyclic. After the elimination of cycles, Bayesian attack graphs model causal relationships between vulnerabilities in the same way as Bayesian networks (Section 3.2.3) Bayesian attack graphs were first proposed by Liu and Man in order to analyze network vulnerability scenarios with the help of Bayesian inference methods in 2005 [174]. Therefore the formalism advances computational methods in security where uncertainty is considered.

The formalism of Man and Liu is not the only fusion of attack graphs and Bayesian networks. Starting in 2008 a group of researchers including Frigault, Noel, Jajodia, and Wang published a paper on a modified version of Bayesian attack graphs. Their goal was to be able to calculate general security metrics regarding information system networks which also contain probabilistic dependencies [175,176]. Later they extended the formalism, using a second copy of the model as time slice, to also capture dynamic behavior in so called *dynamic Bayesian networks* [177]. In 2012, Poolsappasit et al. revisited the framework to be able to deal with asset identification, system vulnerability, and connectivity analysis, as well as mitigation strategies [27]. All three approaches eliminate cycles that possibly exist in the underlying attack graph. A shortcoming of Liu and Man is that they do not provide a specific procedure on how to achieve this. The group including Frigault refers to a paper on attack graphs [178] which removes cycles through an intricate procedure. Poolsappasit et al. state that they rather analyze "why an attack can happen" and not "how an attack can happen", and therefore "cycles can be disregarded using the monotonicity constraint" mentioned in [179].

Since Bayesian attack graphs are cycle-free, evaluation on them can make use of Bayesian inference techniques. For this it is necessary to provide probabilistic information. The three approaches differ in how they compute quantitative values. Liu and Man provide edge probabilities [174], Frigault et al. give conditional probability tables for nodes which are estimated according to the CVSS score [176] and Poolsappasit et al. use (local) conditional probability distributions for nodes [27]. Furthermore, Poolsappasit et al. augment Bayesian attack graphs with additional nodes and values representing hardening measures (defenses). On the augmented structure they propose a genetic algorithm that solves a multiobjective optimization problem of how to assess the risk in a network system and select optimal defenses [27].

The research group including Wang uses a Topological Vulnerability Analysis (TVA) tool [180,181] to create the attack graphs that serve as basis for constructing Bayesian attack graphs. Poolsappasit et al. have developed an unreferenced in-house tool that allows them to compute with conditional probability distributions.

Wang et al. [176,177] state that their work is also based on a paper by An et al. [161], who use Bayesian networks without cycles for modeling risks of violating privacy in a database.

3.2.5. *Compromise graphs*

McQueen et al. introduced *compromise graphs* in 2006 [182]. Compromise graphs are based on directed graphs,[3] and are used to assess the efficiency of various technical security measures for a given network architecture. The nodes of a compromise graph represent the phases of an attack, detailing how a given target can get compromised. The edges are weighted according to the estimated time required to complete the corresponding phase for this compromise. The overall time needed for the attacker to succeed is computed and compared along different defensive settings, providing a metric to assess and compare the efficiency of these different defensive settings.

The formalism has a sound mathematical formalization: a time to compromise (TTC) metric is modeled for each edge as a random process combining three sub-processes. Each of these processes has a different probability distribution (mixing exponential, gamma, and beta-like distributions). The value for the process model parameters are based on the known vulnerabilities of the considered component and the estimated skill of the attacker. A complete description and justification of such a stochastic modeling is provided by the same authors in a previous paper [183]. In compromise graphs, five types of stages, corresponding to the vertices of the graph, are modeled: recognition, breaching the perimeter, penetration, escalation of privilege, damage.

Compromise graphs are used to evaluate the efficiency of security measures, such as system hardening, firewalls or enhanced authentication. This is achieved by comparing the shortest paths (in terms of TTC) of compromise graphs with and without such measures in place.

The approach is illustrated in [182] by modeling attacks on a SCADA system.

Leversage and Byres adopt a very similar approach in [184,185], called state-time estimation algorithm (STEA), directly inspired by McQueen et al. They combine a slightly modified TTC calculation approach with a decomposition of the attack according to the architectural areas of the targeted system. A recent paper by Nzoukou et al. [186] improves the models of McQueen and Leversage even further. The paper proposes to link the mean TTC to the CVSS metric values [187] of specific vulnerabilities, which makes the employment of easily available inputs possible. To derive the overall mean TTC, the results of individual vulnerabilities are then aggregated using Bayesian networks. This allows us to lift the assumption that all attacking steps are independent.

3.2.6. *Enhanced attack trees*

Enhanced attack trees have been introduced by Çamtepe and Yener to support an intrusion detection engine by modeling complex attacks with time dependencies. This model was first described in a technical report [188] in 2006. One year later, corresponding conference publication [189] was published.

In addition to classical OR and AND gates, enhanced attack trees rely on the use of a new gate, the "ordered-AND",

[3] The authors do not state whether these directed graphs are acyclic or not, but the description of compromise graphs and their examples led us to consider compromise graphs as DAGs.

which allows to capture sequential behavior and constraints on the order of attack steps. The model of enhanced attack trees has sound mathematical foundations. Additionally to the formalism description, [189] devises a new technique for detection of attacks. The new technique is based on automata theory and it allows to verify completeness of enhanced attack tree models with respect to the observed attacks.

The quantification capabilities described in [189] are directly related to intrusion detection (probability of a given attack occurring based on a set of observed events). A confidence attribute measured in percent is defined for subgoals as "the chance of reaching the final goal of the attacker when a subgoal is accomplished". It is computed as the ratio of all accomplished events until a subgoal is realized, over all events of the modeled scenario. This attribute aims at supporting an early warning system, supporting decision-making and reaction before actual damages occur. Moreover, [189] introduces an original parameter called "time to live" which allows to express that some steps are only available in a given time window.

In [190], Mishra et al. also make use of ordered-AND operators, referring to [189]. The authors visually describe Stuxnet and similar attacks, but do not use Çamtepe and Yener's rigorous formalization to analyze the models.

3.2.7. *Vulnerability cause graphs*

Vulnerability cause graphs (VCGs) were invented in 2006 by Ardi, Byers, and Shahmehri as a key element of a methodology that supports security activities throughout the entire software development lifecycle [191].

The formalism can be seen as a root cause analysis for security-related software failures, because it relates vulnerabilities with their causes. In a VCG, every node except for one, has an outgoing directed edge. The single node without a successor is called the exit node and represents the considered vulnerability. All other nodes represent causes. The predecessor–successor (parent–child) relationship shows how certain conditions (nodes) might cause other conditions (nodes) to be a concern. In an improved version of VCGs [192], nodes can be simple, compound or conjunctions. Simple nodes represent conditions that may lead to a vulnerability. Compound nodes facilitate reuse, maintenance, and readability of the models. Conjunctions represent groups of two or more nodes. On the contrary, disjunctions occur if a node has two or more predecessors. In this case, the original nodes might have to be considered if either of its predecessors might have to be considered. Finally, if the causes have to follow a certain order, they are modeled as sequences of nodes. To construct a VCG, the exit node is used as a starting point and refined with causes.

In VCGs, nodes can be annotated as "blocked" if the underlying causes are mitigated. The "blocked" flag allows the user to compute whether the underlying vulnerability (exit node) is also mitigated. VCGs are also equipped with a notion of graph transformations that do not change whether the vulnerability is mitigated or not. The transformations include conversions of conjunctions, reordering of sequences, combination of nodes, conversion to compound nodes, as well as derived transformations.

In [192] the vulnerability CVE-2003-0161, in [193] the vulnerability CVE-2005-2558, and in [194] the vulnerability CVE-2005-3192 is analyzed with the help of VCGs. Furthermore, [195] contains an additional three case studies on common software vulnerabilities which have been performed using VCGs. The SHIELDS project [2] has developed a software tool GOAT [196] to be used in conjunction with VCGs.

VCGs were developed as part of a comprehensive methodology to reduce software vulnerabilities that arise in ad hoc software development. They are the starting point to build security activity graphs (Section 3.3.4). By introducing compound nodes, the inventors of the formalism have created a model that allows different layers of abstraction, which in turn introduced a problematic design decision of how many layers of abstraction are needed.

3.2.8. *Dynamic fault trees for security*

In 2009, Khand [197] adapted several dynamic fault tree [198,199] gates to attack trees, in order to add a dynamic dimension to classical attack trees. The aim of the formalism is similar to that of attack trees (Section 3.1.1).

To overcome limitations of static fault trees, dynamic fault trees [198,199] were invented by Dugan et al. in the early 1990s. They aim at combining the dynamic capacities of Markovian models with the "look and feel" of fault trees. To achieve this, four dynamic gates are used: the "priority-AND" (PAND), the "sequence gate" (SEQ), the "functional dependency gate" (FDEP), and the "cold spare gates" (CSP). Khand reuses directly the three first gates (although renaming FDEP gates by CSUB, for Conditional Subordination, gates), leaving out the CSP gates. The PAND gate reaches a success state if all of its input are realized in a pre-assigned order (from left to right in the graphical notation). The SEQ gate allows to model that a series of events occurs in a particular order (from left to right in the graphical notation). Once all the input events are realized, the gate is verified. The CSUB gate models the need of the realization of a trigger event to allow a possible realization of others events. Dynamic fault trees combine dynamic gates with classical logical gates (AND, OR). Dynamic gates are formally defined with truth tables in [197], and by Markov processes in the general definitions of dynamic fault trees from the safety literature [198,199] (although the description is still incomplete [200]).

There is no quantification aspects developed in [197]. In safety studies, quantifications associated with dynamic fault trees are usually made using Markovian analysis techniques; those might be used here as well, although nothing is said about computational aspects.

The paper by Khand does not specify which tool to use in order to treat the models, but several tools exist for dynamic fault trees in the reliability area, e.g., Galileo [201].

The work of Khand, and especially the use of dynamic gates, has inspired Ivanc and Klobučar to propose the enhanced structural model for attack analysis and education, that is able to reflect the reality better than a pure AND–OR tree [202].

3.2.9. *Serial model for multi-parameter attack trees*

In 2010, the parallel model for multi-parameter attack trees (Section 3.1.5) has been extended by adding a temporal order

on the set of elementary attacks [134]. This new methodology is called *serial model for multi-parameter attack trees* and was studied further in [104,136] and [203].

The model described in [104] and [136] assumes that an adversary performs the attacks in a given prescribed order. In [203], the authors introduce so called fully-adaptive adversary model, where an attacker is allowed to try atomic attacks in an arbitrary order which is not fixed in advance and can be modified based on the results of the previous trials. In both cases, the serial approach allows for a more accurate modeling of an attacker's behavior than the parallel approach. In particular, the attacker can skip superfluous elementary attacks and base his decisions on success or failure of the previously executed elementary attacks.

In [134], an efficient algorithm for computing an attacker's expected outcome assuming a given order of elementary attacks is provided. Taking temporal dependencies into account allows the attacker to achieve better expected outcome than when the parallel model (Section 3.1.5) is used. As remarked in [135], finding the best permutation of the elementary attacks in the serial model for multi-parameter attack trees may turn computing the optimal expected outcome into a super-exponential problem. In [136], Niitsoo proposed a decision-theoretical framework which makes possible to compute the maximal expected outcome of a goal oriented attacker in linear time. In [203], Buldas and Stepanenko propose a game theoretical framework to compute upper bounds of the utility of fully-adaptive adversaries. Inspired by the upper bound concept introduced in [203], the authors of [137] propose a new fully adaptive computational model for attack trees. This model allows the adversary to repeat atomic attacks that have failed and to continue attacking even after having been caught. The paper introduces methods to compute a precise value of the adversarial utility and an approximation of the utility upper bound.

A prototype computer tool supporting the security analysis using the serial model of multi-parameter attack trees has been implemented [138] and described in [139].

A thorough comparison of the parallel and the serial model for multi-parameter attack trees has been given in the Ph.D. thesis of Jürgenson [104]. Baca and Petersen mention that in order to use parameterized attack trees, the user needs to have a good understanding of the motivations of the attacker [52]. To overcome this difficulty cumulative voting is used in countermeasure graphs (Section 3.3.7).

3.2.10. *Improved attack trees*

Improved attack trees aim at dealing with security risks that arise in space-based information systems. They were proposed by Wen-ping and Wei-min [204] in 2011 to more precisely describe attack on the information transmitting links, acquisitions systems, and ground-based supporting and application systems.

The formalism is based on attack trees and explicitly incorporates the use of the sequential AND operator. It is not defined in a formal way. Improved attack trees rely heavily on the description by Schneier and only detail how to specifically compute the system risk.

Improved attack trees provide a specific formula to evaluate a risk value for each leaf node. Starting from these risk values, the risk rate and the risk possibility are computed and multiplied to compute the overall system risk. The formulas distinguish between OR, AND and sequential AND nodes.

3.2.11. *Time-dependent attack trees*

In 2014, Arnold et al. introduced a novel model for attack trees, that we refer to as *time-dependent attack trees*. The goal of this new computational framework is to evaluate the probability of an attack as a function of time [205].

The model of Arnold et al. improves upon previously proposed, time-abstract analysis techniques, such as the standard bottom-up algorithm, which only consider the probability of an attack taking place eventually. Time-dependent attack trees make use of standard AND and OR connectors. In addition, they also allow for SEQ connectors (sequential AND) that encode the order in which conjunctively connected actions need to be performed. The model is formally defined.

Every leaf of a time-dependent attack tree is annotated with a cumulative distribution function (CDF) representing the time needed for the corresponding attack step to be successful. The CDF corresponding to the entire attack tree is then derived by composing the CDFs in the leaves with maximum (for AND nodes), minimum (for OR nodes), and convolution (for SEQ nodes) operations along the tree structure. In general, it is fairly complex to compose the distributions, however, the authors of [205] solved this problem by transforming the attack tree into an acyclic phase-type distribution (APH) expression. APH expressions can be efficiently minimized (compressed) and analyzed by model checkers. The output of such analysis is a CDF of the probability of success over time for the entire attack scenario.

A method to generate and manipulate acyclic phase-type distribution representations, together with the compression algorithm have been implemented in a tool suite called `APHzip`. `APHzip` is wrapped in a web-based interface and is accessible on-line [206]. The effectiveness of the approach presented in this section has been illustrated on three toy case studies that are described in [205]. They demonstrate that the algorithm implemented in `APHzip` yields significant state space compressions so that even complex scenarios can be analyzed efficiently.

3.3. *Static modeling of attacks and defenses*

3.3.1. *Anti-models*

Anti-models [207] have been introduced by van Lamsweerde et al. in 2003. They are closely related to AND–OR goal-refinement structures [208] (sometimes called goal models) used for goal analysis in requirements engineering. Anti-models extend such AND–OR goal-refinement structures with the possibility to model malicious and intentional obstacles to security goals, called anti-goals. They can be used to generate subtle attacks, discard non-realizable or unlikely ones, and derive more effective customized resolutions.

In [207] and later in an extended version [117], van Lamsweerde et al. provide a six steps procedure for a systematic construction of anti-models. First, anti-goals, representing an attacker's goals, are obtained by negating confidentiality, privacy, integrity, availability, authentication or non-repudiation

requirements. For each anti-goal, the questions "who" and "why" are asked to identify potential classes of attackers and their higher-level anti-goals. An AND–OR refinement process is then applied to reach terminal anti-goals that are realizable by the attackers. The resulting AND–OR anti-models relate "attackers, their anti-goals, referenced objects and anti-operations (necessary to achieve their anti-goals) to the attackees, their goals, objects, operations, and vulnerabilities". The construction of anti-models is only informally presented in [207]. Formal techniques developed for AND–OR goal-refinement structures (such as refinement obstacle trees) [208] can be used for the generation and analysis of anti-models. In particular, real-time temporal logic can be employed to model anti-goals as sets of attack scenarios. After identifying possible anti-goals, countermeasures expressed as epistemic extensions of real-time temporal logic operators are selected based on severity or likelihood of the corresponding threat and non-functional system goals that have been identified earlier. Possible resolutions tactics, inspired by solutions proposed for the analysis of non-functional requirements in software engineering, are described in [208] and [117]. Applying resolution operators yields new security goals to be integrated in the model. These new goals are then again refined with the help of AND–OR structures. These, in turn, may require a new round of anti-model construction and analysis.

Anti-models do not include quantitative analysis of security goals or anti-goals.

3.3.2. *Defense trees*

Defense trees[4] are attack trees where leaf nodes are decorated with a set of countermeasures. They have been introduced by Bistarelli et al. in 2006 [209]. The approach combines qualitative and quantitative aspects and serves general security modeling purposes.

The approach proposed by Bistarelli et al. was a first step towards integrating a defender's behavior into models based on attack trees. The analysis methodology for defense trees proposed in [209] and [47] uses rigorous and formal techniques, such as calculation of economic indexes and game theoretical solution concepts. However, the model itself is only introduced verbally and a formal definition is not given.

In [209], the return on attack (ROA) and return on investment (ROI) indexes are used for quantitative analysis of defense trees from the point of view of an attacker and a defender, respectively. The calculation of ROI and ROA is based on the following parameters: costs, impact, number of occurrences of a threat and gain. The indexes provide a useful method to evaluate IT security investments and to support the risk management process. In [47], game theoretical reasoning was introduced to analyze attack–defense scenarios modeled with the help of defense trees. In this paper, a defense tree represents a game between two players: an attacker and a defender. The ROI and ROA indexes, are used as utility functions and allow to evaluate the effectiveness and the profitability of countermeasures. The authors of [47] propose using Nash equilibria to select the best strategy for the players.

In [210], defense trees have been extended to so called *CP-defense trees*, where modeling of preferences between countermeasures and actions is possible. Transforming CP-defense trees into answer set optimization (ASO) programs, allows to select the most suitable set of countermeasures, by computing the optimal answer set of the corresponding ASO program. Formalisms such as attack–defense trees (Section 3.3.6), and attack countermeasure trees (Section 3.3.5) extended defense trees by allowing defensive actions to be placed at any node of the tree and not only at the leaf nodes.

3.3.3. *Protection trees*

Protection trees are a tree-based formalism which allow a user to allocate limited resources towards the appropriate defenses against specified attacks. The methodology was invented by Edge et al. in 2006, in order to incorporate defenses in the attack tree methodology [48].

Protection trees are similar to attack trees since both decompose high level goals into smaller manageable pieces by means of an AND–OR tree structure. The difference is that in protection tree the nodes represent protections. A protection tree is generated from an already established attack tree by finding a protection against every leaf node of the attack tree. Then the attack tree is traversed in a bottom-up way and new protection nodes are added to the protection tree if the protection nodes do not already cover the parent attack node.

The AND–OR structure of protection trees is enriched with three metrics, namely probability of success, financial costs, and performance costs on which the standard bottom-up approach is applied [48,211,99]. In [212], an additional metric, the impact, helps to further prioritize where budget should be spent.

The formalism has been investigated in case studies on how the US Department of Homeland Security can allocate resources to protect their computer networks [48], how an attack on an online banking system can be mitigated cost-efficiently [211], how to cheaply protect against an attack on computer and RFID networks [212] as well as a mobile ad hoc network [99]. When evaluating which defenses to install, the authors propose to first prune the tree according to the attacker's assumed capabilities. A larger, more applied case study to "evaluate the effectiveness of attack and protection trees in documenting the threats and vulnerabilities present in a generic Unmanned Aerial Systems (UAS) architecture" was performed by Cowan et al. [213].

In [211] a slightly different algorithm for the creation of a protection tree was proposed. Here a designer starts by finding defenses against the root of an attack tree instead of the leaves, as in [48,99]. An approach similar to protection trees has been proposed in [214] to deal with the problem of threat modeling in software development. The paper uses so called identification trees to identify threats in software design and introduces the model of mitigation trees to describe countermeasures for identified threats. Despite an obvious modeling analogy between protection trees and mitigation trees, no connection between the two models has been made explicit in the literature.

[4] Papers by Bistarelli et al. use British English, thus originally, the name of their formalism is *defence trees*.

3.3.4. *Security activity graphs*

In 2006, Ardi, Byers, and Shahmehri introduced a formalism called *security activity graphs* (SAGs). The methodology was invented in order to "improve security throughout the software development process" [191]. SAGs depict possible vulnerability cause mitigations and are algorithmically generated from vulnerability cause graphs (Section 3.2.7).

SAGs are a graphical representation of first order predicate calculus and are based very loosely on ideas from fault tree analysis. In [191] the root of a SAG is associated with a vulnerability, taken from a vulnerability cause graph. The vulnerability mitigations are modeled with the help of activities (leaf nodes). The syntax furthermore consists of AND-gates, OR-gates, and split gates. The AND and OR-gates strictly follow Boolean logic, whereas the split gate allows one activity to be used in several parent activities, essentially creating a DAGs structure. The syntax of SAGs was changed in [215] for a more concise illustration of the models. Split gates no longer appear in the formalism. The functionality that simple activities can be distinguished from compound activities (complex activities that may require further breakdown) was added. Moreover cause references (possible attack points) serve as placeholders for a different SAG associated with a particular cause.

In the SAG model, Boolean variables are attached to the leaves of the SAG. A Boolean variable corresponding to an activity is true when it "is implemented perfectly during software development" otherwise, it is false. Then a value corresponding to the root of the SAG is deduced in a bottom-up fashion according to Boolean logic.

Visual representation of SAGs is supported by SeaMonster [216] and GOAT [196]. Furthermore, SAGs have been used in [215,193] to model the vulnerability CVE-2005-2558 in MySQL that leads to "denial of service or arbitrary code execution".

Even though the model was devised in order to aid the software development cycle, the authors explicitly state that SAGs "lend themselves to other applications such as process analysis". SAGs are the middle step of a broader 3-steps approach for secure software development, with vulnerability cause graphs as a first step, and process component definition as a final step. In 2010 SAGs were replaced by security goal models (Section 3.4.8)

3.3.5. *Attack countermeasure trees*

In 2010, Roy, Kim, and Trivedi proposed *attack countermeasure trees* (ACTs) [217,218] as a methodology for attack and defense modeling which unifies analysis methods proposed for attack trees (Section 3.1.1) with those introduced on defense trees (Section 3.3.2). The main difference of ACTs with respect to defense trees is that in ACTs defensive measures can be placed at any node of the tree. Also, the quantitative analysis proposed for defense trees is extended by incorporating probabilistic analysis into the model. ACTs were first introduced in [218] and then further developed in [61].

ACTs may involve three distinct classes of events: attack events, detection events, and mitigation events. The set of classical AND and OR nodes, as defined for attack trees, is extended with the possibility of using k-out-of-n nodes. Generation and analysis of attack countermeasure scenarios is automated using minimal cut sets (mincuts). Mincuts help to determine possible ways of attacking and defending a system and to identify the system's most critical components.

A rigorous mathematical framework is provided for quantitative analysis of ACTs in [218] and [61]. The evaluation of the ROI and ROA attributes, as proposed for defense trees (Section 3.3.2), has been extended by adding the probability of attack, detection, and mitigation events. The authors of [61] provide algorithms for probability computation on trees with and without repeated nodes. With the help of probability parameters, further metrics, including cost, impact, Birnbaum's importance measure, and risk, are evaluated. The use of the Birnbaum's importance measure (also called reliability importance measure, in the case of fault trees) is used to prioritize defense mechanisms countering attack events. Furthermore, in [61], Roy et al. propose a cubic algorithm to select an optimal set of countermeasures for an ACT. This addresses the problem of state-space explosion that the intrusion response and recovery engine based on attack–response trees (Section 3.4.5) suffers from. Finally, in [219] the problem of selecting an optimal set of countermeasures with and without having probability assignments has been discussed.

The authors of [61] implemented a module for automatic description and evaluation of ACTs in a modeling tool called Symbolic Hierarchical Automated Reliability and Performance Evaluator [220]. This implementation uses already existing algorithms for the analysis of fault trees and extends them with algorithms to compute costs, impact, and risk. Case studies concerning attacks on the Border Gateway Protocol (BGP), SCADA systems, and malicious insider attacks have been performed using ACTs, as described in the Master thesis of Roy [106].

The model of attack countermeasure trees is very similar to attack–defense trees. The main differences between the two models are listed in Section 3.3.6.

3.3.6. *Attack–defense trees*

Attack–defense trees (ADTrees) were proposed by Kordy et al. in 2010 [221]. They allow to illustrate security scenarios that involve two opposing players: an attacker and a defender. Consequently it is possible to model interleaving attacker and defender actions qualitatively and quantitatively. ADTrees can be seen as merging attack trees (Section 3.1.1) and protection trees (Section 3.3.3) into one formalism.

In ADTrees, both types of nodes, attacks and defenses, can be conjunctively as well as disjunctively refined. Furthermore, the formalism allows for each node to have one child of the opposite type. Children of opposite type represent countermeasures. These countermeasures can be refined and countered again. Two sets of formal definitions build the basis of ADTrees: a graph-based definition and an equivalent term-based definition. The graph-based definition ensures a visual and intuitive handling of ADTree models. The term-based representation allows for formal reasoning about the models. The formalism is enriched through several semantics that allow to define equivalent ADTree representations of a scenario [222]. The necessity for multiple semantics is motivated by diverse applications of ADTrees, in particular unification of other attack tree related approaches and suitability for various kinds of computations. In [223], the

authors showed that, for a wide class of semantics (i.e., every semantics induced by a De Morgan lattice), ADTrees extend the modeling capabilities of attack trees without increasing the computational complexity of the model. In [222] the most often used semantics for ADTrees have been characterized by finite axiom schemes, which provides an operational method for defining equivalent ADTree representations. The authors of [224], have established a connection between game theory and graphical security assessment using ADTrees. More precisely, ADTrees under a semantics derived from propositional logics are shown to be equally expressive as two-player binary zero-sum extensive form games.

The standard bottom-up algorithm for quantitative evaluation, formalized for attack trees in [28], has been extended to ADTrees in [222]. This required the introduction of four new operators (two for conjunction and disjunction of defense nodes and two for countermeasure links) [222]. Together with the two standard operators (for conjunctions and disjunctions of attack nodes) and a set of values, the six operators form an attribute domain. Specifying attribute domains allows the user to quantify a variety of security relevant parameters, such as time of attack, probability of defense, scenario satisfiability, and environmental costs. The authors of [222] show that every attribute for which the attribute domain is based on a semi-ring can be evaluated on ADTrees using the bottom-up algorithm. How to properly specify attribute domains in terms of questions in natural language was presented in [225]. Unfortunately, the bottom-up algorithm can only be applied for the evaluation of the probability attribute under the assumption that all actions in the analyzed ADTree are mutually independent. To lift this assumption, the authors of [226] have proposed a framework that integrates the security model of ADTrees with Bayesian networks and makes possible the computation of the probability of an attack–defense scenario in the presence of dependencies.

An extensive case study on an existing, real-life RFID goods management system was performed by academic and industrial researchers with different backgrounds [20]. The case study resulted in specific guidelines about the use of attributes on ADTrees. A software tool, called the ADTool [227,228], supporting the attack–defense tree methodology, has been developed as one of the outcomes of the ATREES and the TREsPASS projects [5,3]. The main features of the tool are easy creation, efficient editing, and quantitative analysis of ADTrees [229]. Since from a formal perspective, attack trees (Section 3.1.1), protection trees (Section 3.3.3), and defense trees (Section 3.3.2) are instances of attack–defense trees, the ADTool also supports all these formalisms. For an exhaustive overview of the research results related to ADTrees, we refer to the Ph.D. thesis of Schweitzer [114].

Finally, ADTrees can be seen as a natural extension of defense trees (Section 3.3.2), where defenses are only allowed as leaf nodes. The ADTree formalism is quite similar to attack countermeasure trees (Section 3.3.5), however, there exist a couple of fundamental differences between the two models. On the one hand, in ADTrees defense nodes can be refined and countered, which is not possible in attack countermeasure trees. On the other hand, attack countermeasure trees distinguish between detection and mitigation events which are both modeled with defense nodes in ADTrees. Another difference is that attack countermeasure trees are well suited to compute specific parameters, including probability, return on investment (ROI) and return on attack (ROA). ADTrees, in turn, focus on general methods for attribute computation. A different formalism, also called attack–defense trees, was used by Du et al. in [230] to perform a game-theoretic analysis of Vehicular ad hoc network security by utilizing the ROA and ROI utility functions. Despite sharing the same name with the formalism introduced in [221], the attack–defense tree approach used in [230] is built upon defense trees (Section 3.3.2) and does not contain the possibility to refine countermeasures. Moreover it does not consider any formal semantics.

3.3.7. *Countermeasure graphs*

Countermeasure graphs provide a DAG-based structure for identification and prioritization of countermeasures. They were introduced by Baca and Petersen [52] in 2010 as an integral part of the "countermeasure method for security" which aims at simplifying countermeasure selection through cumulative voting.

To build the graphical model, actors, goals, attacks, and countermeasures are identified. Actors are the ones that attack the system, goals explain why actors attack a system, attacks detail how the system could get attacked and countermeasures describe how attacks could be prevented. When the representing events are related, edges are drawn between goals and actors, actors and attacks, as well as between attacks and countermeasures. More specifically, an edge is drawn between a goal and an actor if the actor pursues the goal. An edge is inserted between an actor and an attack, if the actor is likely to be able to execute the attack. Finally, an edge is drawn between an attack and a countermeasure if the countermeasure is able to prevent the attack. Priorities are assigned to goals, actors, attacks, and countermeasures according to the rules of hierarchical cumulative voting [231]. The higher the assigned priority, the higher is the threat level of the corresponding event.

With the help of hierarchical cumulative voting [231] the most effective countermeasures can be deduced. Clever normalization and the fact that countermeasures that prevent several attacks contribute more to the final result than isolated countermeasures guarantee that the countermeasure with the highest computed value is most efficient and should therefore be implemented.

The methodology is demonstrated on an open source system, a first person shooter called Code 43 [52].

3.4. *Sequential modeling of attacks and defenses*

3.4.1. *Insecurity flows*

In 1997, Moskowitz and Kang described a model called *insecurity flows* to support risk assessment [232]. It combines graph theory and discrete probability theory, offering both graphical representation and quantification capabilities to analyze how an "invader can penetrate through security holes to various protective security domains". This analysis aims at identifying the most vulnerable paths and the most appropriate security measures to eliminate the vulnerabilities of the system.

From a high level perspective, insecurity flows are similar to reliability block diagrams [233] used in reliability engineering. The source corresponds to the starting point of the attacker, the sink corresponds to the objective of the attacker, and the asset under protection. An insecurity flow diagram is a circuit connecting security measures, as serial or in parallel, from the sink to the source. Serial nodes must be passed by the attacker one after another, whereas for parallel nodes, only one out of n must be passed to continue on the path to the sink. The graph is used to identify insecurity flows and quantify them using probabilistic calculations. The paper provides a sound description of the formalism and the associated quantifications.

Based on the circuit, the probability that the insecurity flow can pass through the modeled security measures of a given system or architecture can be computed. Probability computation formulas for simple serial and parallel patterns are provided, whereas reduction formulas are proposed for more elaborated circuits (decomposing them into the simple patterns). Several defensive architectures can be compared along this metric.

3.4.2. *Intrusion DAGs*

Intrusion DAGs (I-DAGs) have been introduced by Wu et al. [234] as the underlying structure for attack goals representation in the Adaptive Intrusion Tolerant System, called ADEPTS in 2003. The global goal of ADEPTS is to localize and automatically respond to detected, possibly multiple, and concurrent intrusions on a distributed system.

I-DAGs are directed acyclic graphs representing intrusion goals in ADEPTS. I-DAGs are not necessarily rooted DAGs, i.e., they may have multiple roots. The nodes of an I-DAG represent (sub-)goals of an attack and can be associated with an alert from the intrusion detection framework described in [235]. A goal represented by a node can only be achieved if (some of) the goals of its children are achieved. To model the connection, I-DAGs use standard AND and OR refinement features similar to the refinements in attack trees. Each node stores two information sets: a cause service set (including all services that may be compromised in order to achieve the goal) and an effect service set (including all services that are taken to be compromised once the goal is achieved). The method presented in [234] allows to automatically trigger a response of appropriate severity, based on a value which expresses the confidence that the goal corresponding to a node has been achieved. This provides dynamic aspects to the ADEPTS methodology.

Three algorithms have been developed in order to support automated responses to detected incidents. The goal of the first algorithm is to classify all nodes as candidates for responses as follows. A bottom-up procedure assigns the compromised confidence index to each node situated on the paths between the node representing a detected incident and a root node. Then, a value called threshold is defined by the user and is used by a top down procedure to label the nodes as strong, weak, very weak or non-candidates for potential responses. The second algorithm assigns the response index to nodes. The response index is a real number used to determine the response to be taken for a given node in the I-DAG. Finally, the third algorithm is based on the so called effectiveness index. It is responsible for dynamically deciding which responses are to be taken next. Intuitively, the effectiveness index of a node is reduced for every detected failure of a response action and increased for every successful deployment.

A lightweight distributed e-commerce system has been deployed to serve as a test bed for the ADEPTS tool. The system contained 6 servers and has 26 nodes in the corresponding I-DAG. The results of the experiments and analysis are described in [234].

In [236,237], the authors extend the model of intrusion DAGs to intrusion graphs (I-GRAPHs). The main difference is that, contrary to I-DAGs, I-GRAPHs may contain cycles. Nodes of an I-GRAPH do not need to be independent. All dependencies between the nodes are depicted by the edges between nodes. Additionally to AND and OR refinements, I-GRAPHs also make use of quorum edges. A value called minimum required quorum is assigned to quorum edges and represents the minimal number of children that need to be achieved in order to achieve the parent node.

3.4.3. *Bayesian defense graphs*

In a series of papers starting in 2008, Sommestad et al. construct a Bayesian network for security (Section 3.2.3) that includes defenses to perform enterprise architecture analysis [169,238–240,119]. Their model, explicitly called *Bayesian defense graphs* in [239], is guided by the idea to depict what exists in a system rather than what it is used for [239]. This philosophy was adapted from [241]. Bayesian defense graphs are inspired by defense trees (Section 3.3.2) and therefore add countermeasures to Bayesian networks. As a result, the formalism supports a holistic system view including attack and defense components.

Bayesian defense graphs build upon extended influence diagrams (Section 5.4), including utility nodes, decision nodes, chance nodes, and arcs. Chance nodes and decision nodes are associated with random variables that may assume one of several predefined and mutually exclusive states. The random variables are given as conditional probability tables (or matrices). Utility nodes express the combination of states in chance nodes and decision nodes. Countermeasures, which are controllable elements from the perspective of the system owner, are represented as chance nodes with adapted conditional probability tables. Finally, causal arcs (including an AND or OR label) are drawn between the nodes indicating how the conditional probabilities are related. A strength of Bayesian defense graphs is that they allow to trade-off between collecting as much data as possible and the degree of accuracy of the collected data. Through the use of iterative refinement, it is possible to reduce the complexity of the model [239].

Like all formalisms that involve Bayesian statistics, Bayesian defense graphs use conditional probability tables to answer “How do the security mechanisms influence each other?” and “How do they contribute to enterprise-wide security?” [238]. The authors of [238] exemplify how to compute the expected loss for both the current scenario and potential future scenarios. In [169], a suitable subset of a set of 82 security metrics known as Joint Quarterly Readiness Review (JQRR) metrics has been selected and adapted to Bayesian

Defense graphs. The metrics serve as "a posteriori indicators on the historical success rates of hostile attacks" or "indicate the current state of countermeasures". The formalism can handle causal and uncertainty measurements at the same time, by specifying how to combine the conditional probability tables.

With the help of a software tool for abstract models [241], Bayesian defense graphs were applied by Sommestad et al. to analyze enterprise architectures on numerous occasions. In [240], ongoing efforts on Bayesian defense graphs within the EU research project VIKING [242] are summarized. The methodology is expanded in three follow-up papers that illustrate security assessment based on an enterprise architecture model [238,239] and information flow during a spoofing attack on a server [169]. In [119], a real case study was performed with a power distribution operator to assess the security of wide-area networks (WANs) used to operate electrical power systems. Since the results could not be published the methodology was demonstrated on a fictitious example assessing the security of two communication links with the help of conditional probability tables [119].

A similar but less developed idea of using random variables, defenses, and an inference algorithm to compute the expected cost of an attack is presented by Mirembe and Muyeba [122].

3.4.4. *Security goal indicator trees*

Peine, Jawurek, and Mandel devised *security goal indicator trees* (SGITs) in 2008, in order to support security inspections of software development and documents [243].

A SGIT is a tree which combines negative and positive security features that can be checked during an inspection, in order to see if a security goal (e.g., secure password management) is met. With this objective in mind, "indicators" can be linked in the resulting tree structure by three types of relations: Conditional dependencies are represented by a special kind of edge, Boolean combinations are modeled by OR and AND gates, a "specialization" relation is represented by a UML-like inheritance symbol. Moreover, a notion of "polarity" is defined for each node, attributing positive or negative effect of a given property on security. The definition of SGITs is semi-formal.

The formalism does not support quantitative evaluations.

SGITs are implemented in a prototype tool mentioned in [243]. They are used to formalize security inspection processes for a distributed repository of digital cultural data in an e-tourism application in [19]. The formalism is extended to dependability inspection in [244].

3.4.5. *Attack–response trees*

In 2009, Zonouz, Khurana, Sanders, and Yardley introduced *attack–response trees* (ARTs) as a part of a methodology called response and recovery engine (RRE), which was proposed to automate the intrusion response process. The goal of the RRE is to provide an instantaneous response to intrusions and thus eliminate the delay which occurs when the response process is performed manually. The approach is modeled as a two-player Stackelberg stochastic game between the leader (RRE) and the follower (attacker). Attack–response trees have been used in [245], for the first time. This paper constitutes a part of the Ph.D. thesis of Zonouz [110].

ARTs are an extension of attack trees (Section 3.1.1) that incorporate possible response actions against attacks. They provide a formal way to describe the system security based on possible intrusion and response scenarios for the attacker and the response engine, respectively. An important difference between attack trees and attack–response trees is that the former represent all possible ways of achieving an attack goal and the latter are built based on the attack consequences[5]. In an attack–response tree, a violation of a security property, e.g., integrity, confidentiality or availability, is assigned to the root node (main consequence). Refining nodes represent sub-consequences whose occurrence implies that the parent consequence will take place. Some consequence nodes are then tagged by response nodes that represent response actions against the consequence to which they are connected.

The goal of attack–response trees is to probabilistically verify whether the security property specified by the root of an attack–response tree has been violated, given the sequence of the received alerts and the successfully taken response actions. First, a simple bottom-up procedure is applied in the case when values 0 and 1 are assigned to the leaf nodes. More precisely, when a response assigned to a node v is activated (i.e., is assigned with 1), the values in the subtree rooted in v are reset to 0. Second, [245] also discusses the situation when uncertainties in intrusion detections and alert notifications render the determination of Boolean values impossible. In this case, satisfaction probabilities are assigned to the nodes of attack–response trees and a game-theoretic algorithm is used to decide on the optimal response action. In [246], the RRE has been extended to incorporate both IT system-level and business-level metrics to the model. Here, the combined metrics are used to recommend optimal response actions to security attacks.

The RRE has been implemented on top of the intrusion detection system (IDS) Snort 2.7, as described in [110]. A validation of the approach on a SCADA system use case [245] and a web-based retail company example [246] has shown that this dynamic method performs better than static response mechanisms based on lookup tables. The RRE allows to recover the system with lower costs and is more helpful than static engines when a large number of IDS alerts from different parts of the system are received.

As pointed out in [218], the approach described in this section suffers from the state space explosion problem. To overcome this problem, attack countermeasure trees (Section 3.3.5) have been introduced. Their authors propose efficient algorithms for selecting an optimal set of countermeasures.

3.4.6. *Boolean logic driven Markov process*

Boolean logic driven Markov processes (BDMPs) are a general security modeling formalism, which can also complete generic risk assessment procedures. The formalism was

[5] A reader may notice that what the authors of [245] call "sub-consequences" are in fact the causes of the main consequence.

invented by Bouissou and Bon in 2003 in the safety and reliability area [247] and was adapted to security modeling by Piètre-Cambacédès and Bouissou in 2010 [248,249].[6] Its goal is to find a better trade-off between readability, modeling power, and quantification capabilities with respect to the existing formalisms in general and attack trees in particular.

BDMPs combine the readability of classical attack trees with the modeling power of Markov chains. They change the attack tree semantics by augmenting it with links called triggers. In a first approach, triggers allow modeling of sequences and simple dependencies by conditionally "activating" subtrees of the global structure. The root (top event) of an BDMP is the objective of the attacker. The leaves correspond to attack steps or security events. They are associated to Markov processes, dynamically selected in function of the states of some other leaves. They can be connected by a wide choice of logical gates, including AND, OR, and PAND gates, commonly used in dynamic fault trees (Section 3.2.8). The overall approach allows for sequential modeling in an attack tree-like structure, while enabling efficient quantifications. BDMPs for security are well formalized [249].

Success or realization parameters (mean time to success or to realization) are associated to the leaves, depending on the basic event modeled. Defense-centric attributes can also be added, reflecting detection and reaction capabilities (the corresponding parameters are the probability or the mean-time to detection for a given leaf, and the reduction of chance of success in case of detection). BDMPs for security allow for different types of quantification. These quantifications include the computation of time-domain metrics (overall mean-time to success, probability of success in a given time, ordered list of attack sequences leading to the objectives), attack tree related metrics like costs of attacks, handling of Boolean indicators (e.g., specific requirements), and risk analysis oriented tools like sensibility graphs by attack step or event [251], etc.

The model construction and its analysis are supported by an industrial tool, called KB3 [252]. In [251], implementation issues and user feedback are discussed and analyzed. BDMPs are used in [253–255] to integrate safety and security analyses while [256] develops a realistic use case based on the Stuxnet attack.

In several papers [248,249,251], the authors point out the intrinsic limits of BDMPs to model cyclic behaviors and loops, as well as the difficulties to assign relevant values for the leaves.

3.4.7. *Cyber security modeling language*

The *cyber security modeling language* (CySeMoL) [257–259] has been developed in 2010, by the researchers from the Royal Institute of Technology (KTH) in Sweden. The goal of the language is to estimate the cyber security of enterprise-level system architectures, with a special focus on SCADA systems. This probabilistic relational model [260] specifies how to construct a Bayesian network from an object model.

[6] The original idea was introduced in an abstract by the same authors in 2009 [250]

The big advantage of CySeMoL is that it already includes information on how attacks and defenses relate quantitatively. In order to enable calculations, a user has only to model the system's architecture and some characteristics of the assets involved. The computational procedures of the model assume that the attacker is a professional penetration tester who has fixed, limited time (one work-week) to carry out the attack. Following the work of Sommestad et al., Holm extended CySeMoL into *predictive, probabilistic cyber security modeling language* P^{2}CySeMoL [261]. This extended language has been implemented in the predictive, probabilistic architecture modeling framework [262]. The main improvements introduced in P^{2}CySeMoL are: more flexible and useful computations compared to those implemented in CySeMoL and a possibility of modeling assets, attack steps, and defenses that are common for enterprise architectures which are not necessarily SCADA-related. Furthermore, P^{2}CySeMoL allows the user to manually specify the amount of time that one or more attackers have in order to perform the attack. CySeMoL and P^{2}CySeMoL are formalized using the framework of Bayesian networks.

Literature reviews, analysis of empirical studies, as well as surveys involving domain experts have been conducted to populate CySeMoL and P^{2}CySeMoL models with qualitative information, representing causal relationships between the modeled elements, and quantitative data expressing how likely different attacks are to succeed given the presence or absence of different defenses. This is why computations within CySeMoL or P^{2}CySeMoL can be performed automatically and do not require any personalized input from the user. Based on their computations, the models rank the vulnerabilities of the analyzed systems. In the case of P^{2}CySeMoL, a color encoding on a scale green–yellow–red is used to visualize the obtained quantitative results.

Dedicated tools supporting CySeMoL [263] and P^{2}CySeMoL [264] have been implemented. The practical utility of CySeMoL has been validated in three case studies, focusing on Sweden's three largest electrical power utilities and one of the world's most commonly used electrical power management systems. The sensibility of CySeMoL's assessments has been validated with a variant of the Turing test. P^{2}CySeMoL has been tested in two different case studies, in terms of usability and ease of use. We refer to the Ph.D. thesis of Sommestad [259] for a detailed description of the CySeMoL language, tool, and validation results. An exhaustive presentation of P^{2}CySeMoL and the related literature can be found in the Ph.D. thesis of Holm [261].

3.4.8. *Security goal models*

In 2010, *Security goal models* (SGMs) were formalized by Byers and Shahmehri in order to identify the causes of software vulnerabilities and model their dependencies [265]. They were introduced as a more expressive replacement for attack trees (Section 3.1.1), security goal indicator trees (Section 3.4.4), vulnerability cause graphs (Section 3.2.7), and security activity graphs (Section 3.3.4). The root goal of a SGM corresponds to a vulnerability. "Starting with the root, subgoals are incrementally identified until a complete model has been created" [266].

In SGMs, a goal can be anything that affects security or some other goal, e.g., it can be a vulnerability, a security functionality, a security-related software development activity or an attack. SGMs have two types of goal refinements: one type represents dependencies and one type modeling information flow. Dependency nodes are connected with solid edges (dependence edge) and are depicted by white nodes for contributing subgoals and by black nodes for countering subgoals. Information edges are displayed with dashed edges. The formalism consists of a syntactic domain (elements that make up the model), an abstract syntax (how elements can be combined), a visual representation (used graphical symbols) and a semantic transformation from the syntactic domain to the semantic domain. The syntactic domain consists of the root, subgoals (contributing or counteracting), dependency edges, operators AND and OR that express the connection of dependency edges, annotation connected to nodes by annotation edges, stereotype (usually an annotation about a dependency edge), ports that model information flow, and information edges that connect ports. The abstract syntax is defined in a UML class diagram [266].

It is possible to evaluate whether a security goal was successfully reached or not. To do this, each cause is defined with a logical predicate (true/false). Then the predicates are composed using Boolean logic and taking the information from the information edges into account.

SGM were used in a case study about passive testing vulnerability detection, i.e., examining the traces of a software system without the need for specific test inputs. In a four step testing procedure vulnerabilities are first modeled using SGMs. In the next step, causes are formally defined before SGMs are converted into vulnerability detection conditions (VDC). In the final step vulnerabilities are checked based on the VDCs. In [266] this procedure is performed on the xine media player [267] where an older version contained the CVE-2009-1274 vulnerability. The case study is executed with the help of "TestInv-Code", a program developed by Montimage that can handle VDCs.

In [265], the authors explicitly state that they have defined transformations to and from attack trees VCGs, SAGs, and SGITs so that SGMs can be used with possibly familiar notation. (The transformations, however, were omitted due to space restrictions.)

3.4.9. *Unified parameterizable attack trees*

In 2011, Wang, Whitley, Phan, and Parish introduced *unified parameterizable attack trees*[7] [59]. As the name suggests, the formalism was created as a foundation to unify numerous existing extensions of attack trees (Section 3.1.1). The formalism generalizes the notions of connector types, edge augmentations, and (node) attributes. With the help of these generalizations it is possible to describe other extensions of attack trees as structural extensions, computational extensions or hybrid extensions.

Unified parameterizable attack trees are defined as a 5-tuple, consisting of a set of nodes, a set of edges, a set of allowed connectors (O-AND i.e., a time or priority based AND, U-AND i.e., an AND with a threshold condition and OR), a set of attributes, and a set of edge augmentation structures that allows to specify edge labels. Using this definition, the authors of [59] identify defense trees (Section 3.3.2), attack countermeasure trees (Section 3.3.5), attack–response trees (Section 3.4.5), attack–defense trees (Section 3.3.6), protection trees (Section 3.3.3), OWA trees (Section 3.1.4), and augmented attack trees (Section 3.1.3) as structure-based extensions of attack tree that are covered by unified parameterizable attack trees. They classify multi-parameter attack trees (Sections 3.1.5 and 3.2.9) as a computational extension of attack trees.

The formalism classifies attributes into the categories of "attack accomplishment attributes", "attack evaluation attributes", and "victim system attributes", but does not specify how to perform quantitative evaluations.

Unified parameterizable attack trees are primarily built upon augmented attack trees (Section 3.1.3). In fact, the authors indicate how to instantiate the node attributes, the edge augmentation, and the connector type to obtain an augmented attack tree.

[7] Wang et al. use British English, thus originally, the name of their formalism is *unified parametrizable attack trees*.

4. Summary of the surveyed formalisms

In this section, we provide a consolidated view of all formalisms introduced in Section 3. Tables 2–4 characterize the described methodologies (ordered alphabetically) according to the 13 aspects presented in Table 1. The aspects are grouped into formalism features and capabilities (Table 2), formalism characteristics (Table 3), and formalism maturity and usability factors (Table 4). This tabular view allows the reader to compare the features of the formalisms more easily, it stresses their similarities and differences. Furthermore, the tables support a user in selecting the most appropriate formalism(s) with respect to specific modeling needs and requirements. We illustrate such a support on an exemplary situation.

Example 1. Let us assume that during a risk assessment, analysts want to investigate and compare the efficiency of different defensive measures and controls, with respect to several attack scenarios. Thereto, they need quantitative elements to support the analysis technique they will choose. Furthermore, a software tool and pre-existing use cases are required to facilitate their work. Using the corresponding columns from Tables 2–4 (i.e., attack or defensive, quantification, tool availability, case study) and choosing the formalisms characterized by appropriate values (respectively: both, versatile or specific, industrial or prototype, real(istic)), would help the analysts to pre-selected attack countermeasure trees, attack–defense trees, BDMPs, CySeMoL & P^{2}CySeMoL, intrusion DAGs, and security activity graphs as potential modeling and analysis techniques. The most suitable methodology could then be selected based on more detailed information provided in Section 3. For instance, let us assume that the analysis requires the use of measures for probability of success, the attacker's costs, and the attacker's skills. Checking descriptions of the pre-selected formalisms, given in Section 3, would convince the analysts that security activity graphs and intrusion DAGs would not allow them to compute

Table 2 – Aspects relating to the formalism's modeling capabilities.

Name of formalism	Attack or defense	Sequential or static	Quantification	Main purpose	Extension
Anti-models (Section 3.3.1)	Both	Static	No	Req. eng.	New formalism
Attack countermeasure trees (Section 3.3.5)	Both	Static	Specific	Sec. mod.	Structural Computational
Attack–defense trees (Section 3.3.6)	Both	Static	Versatile	Sec. mod.	Structural Computational
Attack–response trees (Section 3.4.5)	Both	Sequential	Specific	Int. det.	Structural Quantitative
Attack trees (Section 3.1.1)	Attack	Static	Versatile	Sec. mod.	New formalism
Augmented attack trees (Section 3.1.3)	Attack	Static	Specific	Sec. mod.	Structural Computational
Augmented vulnerability trees (Section 3.1.2)	Attack	Static	Specific	Risk	Quantitative
Bayesian attack graphs (Section 3.2.4)	Attack	Sequential	Specific	Risk	Structural Computational
Bayesian defense graphs (Section 3.4.3)	Both	Sequential	Specific	Risk	Structural Computational
Bayesian networks for security (Section 3.2.3)	Attack	Sequential	Specific	Risk	Structural Computational
BDMPs (Section 3.4.6)	Both	Sequential	Versatile	Sec. mod.	Order Time
Compromise graphs (Section 3.2.5)	Attack	Sequential	Specific	Risk	New formalism
Countermeasure graphs (Section 3.3.7)	Both	Static	Specific	Sec. mod.	Structural Computational
Cryptographic DAGs (Section 3.2.1)	Attack	Sequential	No	Risk	New formalism
CySeMoL & P^2CySeMoL (Section 3.4.7)	Both	Sequential	Specific	Risk	New formalism
Defense trees (Section 3.3.2)	Both	Static	Specific	Sec. mod.	Structural Computational
Dynamic fault trees for security (Section 3.2.8)	Attack	Sequential	No	Sec. mod.	Order Time
Enhanced attack trees (Section 3.2.6)	Attack	Sequential	Specific	Int. det.	Order Time
Extended fault trees (Section 3.1.6)	Attack	Static	Specific	Unification	Structural
Fault trees for security (Section 3.2.2)	Attack	Sequential	No	Sec. mod.	Order
Improved attack trees (Section 3.2.10)	Attack	Sequential	Specific	Risk	Structural Computational
Insecurity flows (Section 3.4.1)	Both	Sequential	Specific	Risk	New formalism
Intrusion DAGs (Section 3.4.2)	Both	Sequential	Specific	Int. det.	Structural Computational

(*continued on next page*)

Table 2 (*continued*)

Name of formalism	Attack or defense	Sequential or static	Quantification	Main purpose	Extension
OWA trees (Section 3.1.4)	Attack	Static	Specific	Quantitative	Structural Computational
Parallel model for multi-parameter attack trees (Section 3.1.5)	Attack	Static	Specific	Quantitative	Quantitative Computational
Protection trees (Section 3.3.3)	Defense	Static	Specific	Sec. mod.	New formalism
Security activity graphs (Section 3.3.4)	Both	Static	Specific	Soft. dev.	New formalism
Security goal indicator trees (Section 3.4.4)	Defense	Sequential	No	Soft. dev.	New formalism
Security goal models (Section 3.4.8)	Both	Sequential	Specific	Unification	Structural Computational
Serial model for multi-parameter attack trees (Section 3.2.9)	Attack	Sequential	Specific	Quantitative	Computational Order
Time-dependent attack trees (Section 3.2.11)	Attack	Sequential	Specific	Quantitative	Time Order
Unified parameterizable attack trees (Section 3.4.9)	Both	Sequential	Versatile	Unification	Structural
Vulnerability cause graphs (Section 3.2.7)	Attack	Sequential	Specific	Soft. dev.	Structural Order

the desired quantitative elements. Therefore it would reduce the choice to attack countermeasure trees, attack–defense trees, CySeMoL & P^{2}CySeMoL, and BDMPs. A more thorough investigation of the computational procedures and algorithms described in the referred papers would help the analysts to make the final decision on the formalism that best fits their needs.

5. Alternative methodologies

We close this survey with a short overview of alternative methodologies for graphical security modeling and analysis. The formalisms described here are outside the main scope of this paper, because they were not originally introduced for the purposes of attack and defense modeling or they are not based on the DAG structure. However, for the sake of completeness, we find important to briefly present those approaches as well. The objective of this section is to give pointers to other existing methodological tools for security assessment based on graphical models, rather than to perform a thorough overview of all related formalisms. This explains why the description of the formalisms given here is less complete and structured than the information provided in Section 3.

5.1. *Petri nets for security*

During the mid 1990s, models based on Petri nets have been applied for security analysis [268,269]. In 1994, Kumar and Spafford [268] adopted colored Petri nets for security modeling. They illustrated how to model reference scenarios for an intrusion detection device. Also in 1994, Dacier [269] used Petri nets in his Ph.D. thesis as part of a larger quantification model that describes the progress of an attacker taking over a system. A useful property of Petri nets is their great modeling capability and in particular their ability to take into account the sequential aspect of attacks, the modeling of concurrent action and various forms of dependency. Petri nets are widely used and have various specific extensions. To corroborate this statement, we list a few existing ones. Kumar and Spafford's work relies on *colored Petri nets* [268], Dacier's on *stochastic Petri nets* [269], McDermott's on *disjunctive Petri nets* [270], Horvath and Dörges' on *reference nets* [271], Dalton II et al.'s on *generalized stochastic Petri nets* [272], Pudar et al.'s on *deterministic time transition Petri nets* [273], and Xu and Nygard's on *aspect-oriented Petri nets* [274]. Several articles on Petri nets merge the formalism with other approaches. Horvath and Dörges combine Petri nets with the concept of *security patterns* [271] while Dalton II et al. [272], and more thoroughly Pudar et al. [273], combine Petri nets and attack trees.

In 1994, Dacier embedded Petri nets into a higher level formalism called *privilege graphs*. They model an attacker's progress in obtaining access rights for a desired target [269,275]. In a privilege graph, a node represents a set of privileges and an edge a method for transferring these privileges to the attacker. This corresponds to the exploitation of a vulnerability. The model includes an attacker's "memory" which forbids him to go through privilege states that he has already acquired. In addition, an attacker's "good sense" is modeled

Table 3 – Aspects relating to the formalism's characteristics.

Name of formalism	Structure	Connectors	Formalization
Anti-models (Section 3.3.1)	Tree	AND, OR	Semi-formal
Attack countermeasure trees (Section 3.3.5)	Tree	AND, OR, *k*-out-of-*n*, counter leaves	Formal
Attack–defense trees (Section 3.3.6)	Tree	AND, OR, countermeasures	Formal
Attack–response trees (Section 3.4.5)	Tree	AND, OR, responses	Formal
Attack trees (Section 3.1.1)	Tree	AND, OR	Formal
Augmented attack trees (Section 3.1.3)	Tree	AND, OR	Formal
Augmented vulnerability trees (Section 3.1.2)	Tree	AND, OR	Informal
Bayesian attack graphs (Section 3.2.4)	DAG	AND, OR, conditional probabilities	Formal
Bayesian defense graphs (Section 3.4.3)	DAG	AND, OR, conditional probabilities	Formal
Bayesian networks for security (Section 3.2.3)	DAG	AND, OR, conditional probabilities	Formal
BDMPs (Section 3.4.6)	DAG	AND, OR, PAND, approx. OR, triggers	Formal
Compromise graphs (Section 3.2.5)	Unspecified	None	Formal
Countermeasure graphs (Section 3.3.7)	DAG	Countermeasures	Informal
Cryptographic DAGs (Section 3.2.1)	DAG	Dependence edges	Informal
CySeMoL & P^2CySeMoL (Section 3.4.7)	DAG	Dependence edges, defenses	Semi-formal
Defense trees (Section 3.3.2)	Tree	AND, OR, counter leaves	Semi-formal
Dynamic fault trees for security (Section 3.2.8)	Tree	AND, OR, PAND, SEQ, FDEP, CSP	Informal
Enhanced attack trees (Section 3.2.6)	Tree	AND, OR, ordered-AND	Formal
Extended fault trees (Section 3.1.6)	Tree	AND, OR, merge gates	Formal
Fault trees for security (Section 3.2.2)	Tree	AND, OR, PAND, XOR, inhibit	Informal
Improved attack trees (Section 3.2.10)	Tree	AND, OR, sequential AND	Informal
Insecurity flows (Section 3.4.1)	Unspecified	None	Formal
Intrusion DAGs (Section 3.4.2)	DAG	AND, OR	Semi-formal
OWA trees (Section 3.1.4)	Tree	OWA operators	Formal

(*continued on next page*)

Table 3 (*continued*)

Name of formalism	Structure	Connectors	Formalization
Parallel model for multi-parameter attack trees (Section 3.1.5)	Tree	AND, OR	Formal
Protection trees (Section 3.3.3)	Tree	AND, OR	Informal
Security activity graphs (Section 3.3.4)	DAG	AND, OR, split gate	Semi-formal
Security goal indicator trees (Section 3.4.4)	Tree	AND, OR, dependence edge, specialization edge	Semi-formal
Security goal models (Section 3.4.8)	DAG	AND, OR, dependence edge, information edge	Formal
Serial model for multi-parameter attack trees (Section 3.2.9)	Tree	AND, OR, ordered leaves	Formal
Time-dependent attack trees (Section 3.2.11)	DAG	AND, OR, SEQ	Formal
Unified parameterizable attack trees (Section 3.4.9)	Tree	AND, OR, PAND, time-based AND, threshold AND	Formal
Vulnerability cause graphs (Section 3.2.7)	DAG	AND, OR, sequential AND	Informal

which prevents him from regressing. In [276], Dacier et al. proposed to transform a privilege graph into a Markov chain corresponding to all possible successful attack scenarios. The method has been applied to help system administrators to monitor the security of their systems.

In [277], Zakrzewska and Ferragut presented a model extending Petri nets in order to model real-time cyber conflicts. This formalism is able to represent situational awareness, concurrent actions, incomplete information, and objective functions. Since it makes use of stochastic transitions, it is well suited to reason about stochastic non-controlled events. The formalism is used to run simulations of cyber attacks in order to experimentally analyze cyber conflicts. The authors also performed a comparison of their *extended Petri nets* model with other security modeling techniques. In particular, they showed that extended Petri nets are more readable and more expressive than attack graphs, especially with respect to the completeness of the models.

5.2. *Attack graphs*

The term *attack graph* has been first introduced by Phillips and Swiler [278,279] in 1998, and has extensively been used ever since. The nodes of an attack graph represent possible states of a system during the attack. The edges correspond to changes of states due to an attacker's actions. An attack graph is generated automatically based on three types of inputs: attack templates (generic representations of attacks including required conditions), a detailed description of the system to be attacked (topology, configurations of components, etc.), and the attacker's profile (his capability, his tools, etc.). Quantifications, such as average probabilities or time to success, can be deduced by assigning weights to the edges and by finding shortest paths in the graph.

Starting in 2002, Sheyner et al. [280,281] made extensive contributions to popularize attack graphs by associating them with model checking techniques. To limit the risk of combinatorial explosion, a large number of methods were developed. Ammann et al. [179] restricted the graphs by exploiting a monotony property, thereby eliminating backtracking in terms of privilege escalation. Noel, Jajodia, and others [282,180] took configuration aspects into account. A complete state of the art concerning the contributions to the field between 2002 and 2005 can be found in [283]. In 2006, Wang et al. introduced a relational model for attack graphs [284]. The approach facilitates interactive analysis of the models and improves its performance. Ou et al. [285] optimized the generation and representation of attack graphs by transforming them into *logical attack graphs* of polynomial size with respect to the number of components of the computer network analyzed. During the same year, Ingols et al. [286] proposed *multiple-prerequisite graphs*, which also severely reduce the complexity of the graphs. In [287], Mehta et al. proposed an algorithm for the classification of states in order to identify the most relevant parts of an attack graph. In 2008, Malhotra et al. [288] did the same based on the notion of an *attack surface* described in [289]. The vast majority of the authors mentioned have also worked on visualization aspects [290–293]. Kotenko and Stepashkin [294] described a complete software platform for implementing concepts and metrics of attack graphs. On a theoretical level, Braynov and Jadliwala [295] extended the model to several attackers.

Starting in 2003, the problem of quantitative assessment of the security of networked systems using attack graphs has been extensively studied [282,296–298,178]. The work presented in [282] and [296] focuses on minimal cost of removing vulnerabilities in hardening a network. In [297], the authors introduced a metric, called attack resistance, which is used to

Table 4 – Aspects related to the formalism's maturity and usability.

Name of formalism	Tool availability	Case study	External use	Paper count	Year
Anti-models (Section 3.3.1)	No	No	No	3	2006
Attack countermeasure trees (Section 3.3.5)	Prototype	Real(istic)	No	4	2010
Attack–defense trees (Section 3.3.6)	Prototype	Real(istic)	Collaboration	8	2010
Attack–response trees (Section 3.4.5)	Prototype	Toy case study	No	3	2009
Attack trees (Section 3.1.1)	Commercial	Real(istic)	Independent	> 100	1991
Augmented attack trees (Section 3.1.3)	No	Real(istic)	Independent	6	2005
Augmented vulnerability trees (Section 3.1.2)	No	Real(istic)	Independent	3	2003
Bayesian attack graphs (Section 3.2.4)	Commercial	Toy case study	Independent	10	2005
Bayesian defense graphs (Section 3.4.3)	Prototype	Real(istic)	No	5	2008
Bayesian networks for security (Section 3.2.3)	Commercial	Real(istic)	Independent	14	2004
BDMPs (Section 3.4.6)	Commercial	Real(istic)	Independent	6	2010
Compromise graphs (Section 3.2.5)	No	Real(istic)	Collaboration	3	2006
Countermeasure graphs (Section 3.3.7)	No	Toy case study	No	1	2010
Cryptographic DAGs (Section 3.2.1)	No	No	No	1	1996
CySeMoL & P^2CySeMoL (Section 3.4.7)	Prototype	Real(istic)	No	6	2010
Defense trees (Section 3.3.2)	No	No	No	3	2006
Dynamic fault trees for security (Section 3.2.8)	No	No	No	1	2009
Enhanced attack trees (Section 3.2.6)	No	No	No	1	2007
Extended fault trees (Section 3.1.6)	No	No	No	1	2007
Fault trees for security (Section 3.2.2)	Commercial	Real(istic)	Independent	3	2003
Improved attack trees (Section 3.2.10)	No	No	No	1	2011
Insecurity flows (Section 3.4.1)	No	No	No	1	1997
Intrusion DAGs (Section 3.4.2)	Prototype	Real(istic)	No	2	2003
OWA trees (Section 3.1.4)	No	No	No	2	2005

(*continued on next page*)

Table 4 (*continued*)

Name of formalism	Tool availability	Case study	External use	Paper count	Year
Parallel model for multi-parameter attack trees (Section 3.1.5)	Prototype	Real(istic)	Collaboration	5	2006
Protection trees (Section 3.3.3)	No	Toy case study	No	4	2006
Security activity graphs (Section 3.3.4)	Prototype	Real(istic)	No	2	2006
Security goal indicator trees (Section 3.4.4)	Prototype	Real(istic)	No	3	2008
Security goal models (Section 3.4.8)	No	Real(istic)	No	2	2010
Serial model for multi-parameter attack trees (Section 3.2.9)	Prototype	No	No	4	2010
Time-dependent attack trees (Section 3.2.11)	Prototype	Toy case study	No	1	2014
Unified parameterizable attack trees (Section 3.4.9)	No	No	No	1	2011
Vulnerability cause graphs (Section 3.2.7)	Commercial	Real(istic)	Independent	4	2006

compare the security of different network configurations. The approach was then extended in [298] into a general abstract framework for measuring various aspects of network security. In [178], Wang et al. introduced a metric incorporating probabilities of the existence of the vulnerabilities considered in the graph.

In his Master thesis, Louthan IV [299] proposed extensions to the attack graph modeling framework to permit modeling continuous, in addition to discrete, system elements and their interactions.

In [178], Wang et al. addressed the problem of likelihood quantification of potential multi-step attacks on networked environments, that combine multiple vulnerabilities. They developed an attack graph-based probabilistic metric for network security and proposed heuristics for efficient computation. In [175], Noel et al. used attack graphs to understand how different vulnerabilities can be combined to form an attack on a network. They simulated incremental network penetration and assessed the overall security of a network system by propagating attack likelihoods. The method allows to give scores to risk mitigation options in terms of maximizing security and minimizing cost. It can be used to study cost/benefit trade-offs for analyzing return on security investment.

Dawkins and Hale [300] developed a concept similar to attack graphs called *attack chains*. The model is based on a deductive tree structure approach but also allows for inductive reasoning using *goal-inducing attack chains*, to extract scenarios leading to a given aim. These models are also capable of generating attack trees, which may be quantified by conventional methods. Aspects concerning software implementation are described in [301].

In [302], Samarji et al. introduce a new formal approach called *simultaneous attacks graphs*. Contrary to previous work on attack graphs restricted to individual attacks, as in [280], simultaneous attacks graphs model individual, coordinated, as well as concurrent attacks. Samarji et al. show how to automatically generate simultaneous attacks graphs using the situation calculus formalism [303]. The objective is to support response systems in the estimation of risk inferred from simultaneous ongoing attacks, and to choose appropriate responses.

5.3. *Approaches derived from UML diagrams*

We start this section with a short description of two formalisms derived from UML diagrams, namely *abuse cases* of McDermott and Fox [304] and *misuse cases* of Sindre and Opdahl [305–309] which were later extended by Røstad in [310]. These techniques are not specifically intended to model attacks but rather to capture threats and abusive behavior which have to be taken into account when eliciting security requirements (for misuse cases) as well as for design and testing (for abuse cases). The flexibility of misuse and abuse cases allows for expressive graphical modeling of attack scenarios without mathematical formalization that supports quantification.

In [311], Firesmith argues that misuse and abuse cases are "highly effective ways of analyzing security threats but are inappropriate for the analysis and specification of security requirements". The reasoning is that misuse cases focus on how misusers can successfully attack the system. Thus they often model specific architectural mechanisms and solutions, e.g., the use of passwords, rather than actual security requirements, e.g., authentication mechanisms. To specify security requirements, he suggested to use *security use cases*. Security use cases focus on how an application achieves its goals. According to Firesmith, they provide "a highly-reusable

way of organizing, analyzing, and specifying security requirements" [311].

Diallo et al. presented a comparative evaluation of the common criteria [312], misuse cases, and attack trees [313]. Opdahl and Sindre [314] compared usability aspects and modeling features of misuse cases and attack trees. UML-based approaches can be combined with other types of models. The combination of misuse cases and attack trees appears not only to be simple but also useful and relevant [315,316]. In [317], Kárpáti et al. adapted use case maps to security as *misuse case maps*. Katta et al. [318] combined *UML sequence diagrams* with misuse cases in a new formalism called *misuse sequence diagrams*. A misuse sequence diagram represents a sequence of attacker interactions with system components and depicts how the components were misused over time by exploiting their vulnerabilities. The authors of [318] performed usability and performance comparison of misuse sequence diagrams and misuse case maps. In [319], Kárpáti et al. integrated five different representation techniques in a method called *hacker attack representation method* (HARM). The methodologies used in HARM are: attack sequence descriptions (summarizing attacks in natural language), misuse case maps (depicting the system architecture targeted by the attack and visualizing the traces of the exploits), misuse case diagrams (showing threats in relation to the wanted functionality), attack trees (representing the hierarchical relation between attacks), and attack patterns (describing an attack in detail by adding information about context and solutions). Combining such diverse representation techniques has two goals. First, it provides "an integrated view of security attacks and system architecture". Second, the HARM method is especially well suited when different stakeholders, including non-technical people preferring informal representations are involved in modeling a security scenario.

In [320], Sindre adapted UML activity diagrams to security. The resulting *mal-activity diagrams* constitute an alternative for misuse cases when the author considers the latter to be unsuitable. This is for instance the case in situations where a large numbers of interactions need to be specified within or outside a system. Case studies mainly concern social engineering attacks [321].

5.4. *Isolated models*

In this section we gather a number of isolated models. Most of them contain cycles and therefore are outside of the main scope of this paper. However, we mention them because they build upon one of the formalisms described in Section 3.

The *stratified node topology* was proposed by Daley et al. [322] as an extension of attack trees, in 2002. The formalism consists of a directed graph which is aimed at providing a context sensitive attack modeling framework. It supports incident correlation, analysis, and prediction and extends attack trees by separating the nodes into three distinct classes based on their functionality: event-level nodes, state-level nodes and top-level nodes. The directed edges between the nodes are classified into implicit and explicit links. Implicit links allow individual nodes to imply other nodes in the tree; explicit links are created when an attack provides a capability to execute additional nodes, but does not actually invoke a new instance of a node. As in attack trees, the set of linked nodes can be connected disjunctively as well as conjunctively. In comparison with attack trees, the authors drop the requirement of a designated root node, along with the requirement that the graphs have to be acyclic. Due to the functional distinction of the nodes, the stratified node topology can keep the vertical ordering, even if the modeled scenario is cyclic.

In 2010, Abdulla et al. [51] described a model called *attack jungles*. When trying to use attack trees as formalized by Mauw and Oostdijk in [28] to illustrate the security of a GSM radio network, the authors of [51] encountered modeling problems related to the presence of cycles as well as analysis problems related to reusability of nodes in real life scenarios. This led them to propose attack jungles, which extend attack trees with multiple roots, reusable nodes, and cycles that allow for modeling of attacks which depend on each other. Attack jungles are formalized as multigraphs and their formal semantics extend the semantics based on multisets proposed in [28]. In order to find possible ways of attacking a system, a backwards reachability algorithm for the analysis of attack jungles was described. Moreover, the notion of an attribute domain for quantitative analysis, as proposed for attack trees in [28], is extended to fit the new structure of attack jungles. By dividing attack components (nodes) into reusable and not reusable ones, it is possible to better analyze realistic scenarios. For instance, in an attack jungle it is possible to indicate that a component used once can be reused multiple times without inducing any extra cost.

Extended influence diagrams [323] form another related formalism which is not based on a DAG structure. Extended influence diagrams are built upon influence diagrams, introduced by Matheson and Howard in the 1960s [324], which, in turn, are an extension of Bayesian networks. Influence diagrams are used to provide a high-level visualization of decision problems under uncertainty [325]. Extended influence diagrams allow to model the relationships between decisions, events, and outcomes of an enterprise architecture. They employ the following three types of nodes: ellipses which represent events (also known as chance nodes), rectangles which depict decision nodes and diamonds which represent utility nodes (or outcomes). In addition the formalism allows to specify how a node is defined, how well it can be controlled, and how the nodes relate to each other. The latter is achieved using different types of edges. Moreover, transformation rules between graphs govern switching between different levels of abstraction of a scenario (expanding and collapsing). The rules also ensure that graphs do not contradict each other. In [326], the authors show how to elicit knowledge from scientific texts, generating extended influence diagrams and in [240] the authors outline how extended influence diagrams can be used for cyber security management.

6. Conclusion

This work presents a complete and methodical overview of DAG-based techniques for modeling attack and defense scenarios. Some of the described methodologies have extensively been studied and are widely used to support security and risk assessment processes. Others emerged from specific,

practical developments and have remained isolated methods. This survey provides a structured description of the existing formalisms, gives pointers to related papers, tools, and projects and proposes a general classification of the presented approaches. To classify the formalisms, we have used 13 aspects concerning graphical, formal, and usability characteristics of the analyzed models.

Two general trends can be observed in the field of graphical security modeling: *unification* and *specification*. The objective of the methodologies developed within the first trend is to unify existing approaches and propose general solutions that can be used for analysis of a broad spectrum of security scenarios. The corresponding formalisms are well suited for reasoning about situations involving diversified aspects, such as digital, physical and social components, simultaneously. Such models usually have sound formal foundations and are extensively studied from a theoretical point of view. They are augmented with formal semantics and a general mathematical framework for quantitative analysis. Examples of such models developed within the unification trend are attack–defense trees, unified parameterizable attack trees, multi-parameter attack trees, OWA trees, Bayesian attack graphs, and Bayesian defense graphs.

The second observed trend, i.e., the specification trend, aims at developing methodologies for addressing domain specific security problems. Studied domains include intrusion detection (e.g., attack–response trees, intrusion DAGs), secure software development (e.g., security activity graphs, security goal indicator trees), and security requirements engineering (e.g., anti-models). Formalisms developed within this trend are often based on empirical studies and practical needs. They concentrate on domain specific metrics, such as the *response index*, which is used for the analysis of intrusion DAGs. These approaches often remain isolated and seldom relate to or build upon other existing approaches.

The multitude of methodologies presented in this survey shows that graphical security modeling is a young but very rapidly growing area. Thus, further development is necessary and new directions need to be explored before security assessment can fully benefit from graphical models. One of the research questions which has not yet received enough attention is building graphical models from pre-existing attack templates and patterns. Addressing this question would make automatic model creation possible and replace the tedious, error-prone, manual construction process. It would therefore strongly relieve the industrial sector when building large-scale practical models.

The idea of reusing attack patterns is not new. It has already been mentioned in 2001 by Moore et al. [40]. An excellent initiative was taken by the FP7 project SHIELDS [2], in which the Security Vulnerability Repository Service (SVRS) has been developed. The SVRS is an on-line library of various security models including attack trees [327]. A natural follow-up step is to propose methods for automatic or semi-automatic construction of complex, specific models from general attack or vulnerability patterns. This would require developing algorithms for correct composition and comparison of models, standardizing employed node labels and introducing an agent-based view into the formalisms.

Using security patterns makes threat analysis more efficient and accurate. Generating a general model from existing libraries constitutes a good starting point for further model refinement and analysis. Furthermore, although new technological opportunities arise every day, empirical studies show that most attackers reuse the same attack vectors with little or no modification. Often the same company is attacked several times by an intruder exploiting the same already known vulnerability.

There still exists a gap between theoretical research and practical employment of graphical security models. Tighter interaction between the scientific and industrial security communities would be very beneficial for the future of the field. Having this goal in mind, the *First International Workshop on Graphical Models for Security* (GraMSec) has been set up and took place in April 2014 [328]. The objective of GraMSec is to allow practitioners to better understand the capabilities of theoretical models and give scientists an opportunity to learn what the practical and industrial needs are. Next edition of the workshop has already been planned and it is the organizers' intention to make GraMSec a yearly event providing a platform for the exchange of ideas, collaboration, and result dissemination in the field.

Once a bridge between the scientific and the industrial communities is built, a natural next step will be to include graphical models into standardized and commonly used auditing and risk assessment tools and practices. Due to the sound formal foundations of the graphical models as well as their user friendliness, this would greatly improve the quality and usability of the currently used, mostly table-based, practical risk and auditing methodologies.

Acknowledgments

The authors would like to thank Sjouke Mauw, Marc Bouissou, and Pieter Hartel for their comments on a preliminary version of this survey, which helped them to considerably improve the paper. The research leading to these results has received funding from the *Fonds National de la Recherche Luxembourg* under the grants C08/IS/26 and PHD-09-167 and the European Commission's Seventh Framework Program (FP7/2007–2013) under the grant agreement 318003 (TREsPASS). This publication reflects only the authors' views and the funding bodies are not liable for any use that may be made of the information contained herein.

REFERENCES

[1] B. Kordy, S. Mauw, and W. Pieters (Eds.), 2014. in: Proceedings First International Workshop on Graphical Models for Security, GraMSec 2014, EPTCS, vol. 148, Grenoble, France, 12th April, 2014.

[2] SHIELDS, 2008–2010. SHIELDS: Detecting known security vulnerabilities from within design and development tools, FP7 project, grant agreement 215995.
URL http://www.shields-project.eu/.

[3] TREsPASS, 2012–2016. Technology-supported risk estimation by predictive assessment of socio-technical security, FP7 project, grant agreement 318003.
URL http://www.trespass-project.eu/.

[4] ANIKETOS, 2010–2014. ANIKETOS: Ensuring Trustworthiness and Security in Service Composition, FP7 project, grant agreement 257930. URL http://www.aniketos.eu/.
[5] ATREES, 2009–2012. Attack Trees, project funded by the Fonds National de la Recherche, Luxembourg under grants C08/IS/26 and PHD-09-167.
URL http://satoss.uni.lu/projects/atrees/.
[6] ADT2P, 2014–2017. Attack–defense trees: theory meets practice, project funded by the Fonds National de la Recherche, Luxembourg under grant C13/IS/5809105. URL http://www.fnr.lu/en/Research-Programmes/Funding-by-Call-Type/Projects/Attack-Defense-Trees-Theory-Meets-Practice-ADT2P.
[7] VISPER, 2007–2011. VISPER: The VIrtual Security PERimeter for digital, physical, and organisational security, project funded by the Sentinels programme.
URL http://www.sentinels.nl/en/content/visper.
[8] SESAMO, 2012–2015. Security and safety modelling, ARTEMIS JU Project N. 295354.
URL http://sesamo-project.eu/.
[9] CRUTIAL 2006–2009. CRitical UTility InfrastructurAL resilience, IST-FP6-STREP project, grant agreement 027513. URL http://crutial.rse-web.it.
[10] E.J. Byres, M. Franz, D. Miller, The use of attack trees in assessing vulnerabilities in SCADA systems, in: International Infrastructure Survivability Workshop (IISW'04), Institute of Electrical and Electronics Engineers, Lisbon, 2004, http://blogfranz.googlecode.com/files/SCADA-Attack-Trees-IISW.pdf.
[11] C.-W. Ten, C.-C. Liu, G. Manimaran, 2007. Vulnerability assessment of cybersecurity for SCADA systems using attack trees. in: Proceedings of the IEEE Power Engineering Society General Meeting. Tampa, USA, pp. 1–8.
[12] E. Tanu, J. Arreymbi, 2010. An examination of the security implications of the supervisory control and data acquisition (SCADA) system in a mobile networked environment: an augmented vulnerability tree approach. in: Proceedings of Advances in Computing and Technology, (AC&T) The School of Computing and Technology 5th Annual Conference. University of East London, School of Computing, Information Technology and Engineering, pp. 228–242. URL http://hdl.handle.net/10552/994.
[13] E.L. Lazarus, D.L. Dill, J. Epstein, J.L. Hall, 2011. Applying a reusable election threat model at the county level. in: Proceedings of the 2011 Conference on Electronic voting Technology/Workshop on Trustworthy Elections. EVT/WOTE'11. USENIX Association, Berkeley, CA, USA, pp. 1–14.
[14] A. Buldas, T. Mägi, 2007. Practical Security Analysis of E-Voting Systems. In: [329], pp. 320–335. URL
http://aeolus.ceid.upatras.gr/scientific-reports/2nd_year_reports/practical_e_voting_final.pdf.
[15] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, B. Weyl, 2009. Security requirements for automotive on-board networks. in: 9th International Conference on Intelligent Transport Systems Telecommunications,(ITST). Lille, pp. 641–646.
[16] A. Aijaz, B. Bochow, F. Dötzer, A. Festag, M. Gerlach, R. Kroh, T. Leinmüller, 2006. Attacks on inter vehicle communication systems—an analysis. In: 3rd International Workshop on Intelligent Transportation. pp. 189–194.
[17] T. Tidwell, R. Larson, K. Fitch, J. Hale, 2001. Modeling internet attacks. in: Proceedings of the 2nd IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW '01). West Point, USA, pp. 54–59.
[18] X. Lin, P. Zavarsky, R. Ruhl, D. Lindskog, 2009. Threat modeling for CSRF attacks. in: International Conference on Computational Science and Engineering (CSE'09). vol. 3. pp. 486–491.
[19] C. Jung, F. Elberzhager, A. Bagnato, F. Raiteri, 2010. Practical experience gained from modeling security goals: using SGITs in an industrial project. in: International Conference on Availability, Reliability, and Security (ARES'10). pp. 531–536.
[20] A. Bagnato, B. Kordy, P.H. Meland, P. Schweitzer, Attribute decoration of attack–defense trees, Int. J. Secure Softw. Eng. 3 (2) (2012) 1–35. Special Issue on Security Modeling.
[21] J.-H. Eom, M.-W. Park, S.-H. Park, T.-M. Chung, 2011. A framework of defense system for prevention of insider's malicious behaviors. in: 13th International Conference on Advanced Communication Technology (ICACT'11). pp. 982–987.
[22] K. Reddy, H.S. Venter, M. Olivier, I. Currie, 2008. Towards Privacy taxonomy-based attack tree analysis for the protection of consumer information privacy. in: Proceedings of the 6th Annual Conference on Privacy, Security and Trust (PST'08). New Brunswick, Canada, pp. 56–64.
[23] J.D. Weiss, 1991. A system security engineering process. in: 14th Annual NCSC/NIST National Computer Security Conference. pp. 572–581.
[24] W.E. Vesely, F.F. Goldberg, N.H. Roberts, D.F. Haasl, Fault Tree Handbook Tech. Rep. NUREG-0492, U.S. Regulatory Commission, 1981, URL http://www.nrc.gov/reading-rm/doc-collections/nuregs/staff/sr0492/sr0492.pdf.
[25] C. Salter, O.S. Saydjari, B. Schneier, J. Wallner, 1998. Toward a secure system engineering methodology. in: Proceedings of the 1998 Workshop on New Security Paradigms (NSPW '98). Charlottesville, Virginia, United States, pp. 2–10.
[26] B. Schneier, Attack trees: modeling security threats, Dobb's J. Softw. Tools 24 (12) (1999) 21–29. URL
http://www.ddj.com/security/184414879.
[27] N. Poolsappasit, R. Dewri, I. Ray, Dynamic security risk management using Bayesian attack graphs, IEEE Trans. Dependable and Secure Computing 9 (1) (2012) 61–74.
[28] S. Mauw, M. Oostdijk, Foundations of attack trees, in: D. Won, S. Kim (Eds.), ICISC, in: LNCS., vol. 3935, Springer, 2005, pp. 186–198. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.1056.
[29] R.R. Yager, OWA trees and their role in security modeling using attack trees, Inf. Sci. 176 (20) (2006) 2933–2959.
[30] E.G. Amoroso, Fundamentals of Computer Security Technology, Prentice-Hall, Inc, Upper Saddle River, NJ, USA, 1994, URL http://portal.acm.org/citation.cfm?id=179237.
[31] F. Swiderski, W. Snyder, Threat Modeling, Microsoft Press, Redmond, 2004, URL
http://books.google.lu/books?id=xawLAAAACAAJ.
[32] M. Howard, D. LeBlanc, Writing Secure Code, second ed., Microsoft Press, 2002.
[33] A. Marback, D. Hyunsook, K. He, S. Kondamarri, D. Xu, 2009. Security test generation using threat trees. in: Automation of Software Test, 2009. AST'09. ICSE Workshop on. pp. 62–69.
[34] U.S. Department of Defense (DoD), 1988. Standard practice for system safety. MIL-STD-882D.
[35] P. Ongsakorn, K. Turney, M.A. Thornton, S. Nair, S.A. Szygenda, T. Manikas, 2010. Cyber threat trees for large system threat cataloging and analysis. in: 4th Annual IEEE Systems Conference. pp. 610–615.
[36] J. Steffan, M. Schumacher, 2002. Collaborative attack modeling. in: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'02). Madrid, Spain, pp. 253–259.
[37] B. Schneier, Secrets & Lies: Digital Security in a Networked World, Wiley, Indianapolis, Ind, 2004.
[38] X. Qin, W. Lee, 2004. Attack plan recognition and prediction using causal networks. in: 20th Annual Computer Security Applications Conference. pp. 370–379.

[39] D.M. Kienzle, W.A. Wulf, A practical approach to security assessment, in: Proceedings of the 1997 New Security Paradigms Workshop. NSPW'97, ACM, New York, NY, USA, 1997, pp. 5–16. URL http://doi.acm.org/10.1145/283699.283731.
[40] A.P. Moore, R.J. Ellison, R.C. Linger, Attack modeling for information security and survivability. Technical Note CMU/SEI-2001-TN-001, Carnegie Mellon University, 2001.
[41] R.C. Linger, A.P. Moore, 2001. Foundations for survivable system development: service traces, intrusion traces, and evaluation models. Tech. Rep. CMU/SEI-2001-TR-029, Software Engineering Institute.
[42] R. Vigo, F. Nielson, H.R. Nielson, Automated generation of attack trees, in: CSF'14, IEEE, 2014, in press.
[43] Paul Stéphane, Towards automating the construction & maintenance of attack trees: a feasibility study, in: B. Kordy, S. Mauw, W. Pieters (Eds.), GraMSec, in: EPTCS, vol. 148, 2014, pp. 31–46.
[44] J.N. Whitley, R.C.-W. Phan, J. Wang, D.J. Parish, Attribution of attack trees, Comput. Electr. Eng. 37 (4) (2011) 624–628.
[45] V. Higuero, J.J. Unzilla, E. Jacob, P. Sáiz, D. Luengo, 2004. Application of 'attack trees' technique to copyright protection protocols using watermarking and definition of a new transactions Protocol SecDP (secure distribution protocol). in: Proceedings of the 2nd International Workshop on Multimedia Interactive Protocols and Systems (MIPS'04), LNCS, vol. 3311, Grenoble, France, pp. 264–275.
[46] C. Fung, Y.-L. Chen, X. Wang, J. Lee, R. Tarquini, M. Anderson, R. Linger, 2005. Survivability analysis of distributed systems using attack tree methodology. in: MILCOM. Atlantic City, NJ, pp. 583–589.
[47] S. Bistarelli, M. Dall'Aglio, P. Peretti, Strategic games on defense trees, in: T. Dimitrakos, F. Martinelli, P.Y.A. Ryan, S.A. Schneider (Eds.), FAST, in: LNCS, vol. 4691, Springer, 2006, pp. 1–15. URL http://www.springerlink.com/content/83115122h9007685/.
[48] K.S. Edge, G.C. Dalton II, R.A. Raines, R.F. Mills, Using attack and protection trees to analyze threats and defenses to homeland security, in: MILCOM, IEEE, 2006, pp. 1–7.
[49] V. Saini, Q. Duan, V. Paruchuri, Threat Modeling Using Attack Trees, J. Comput. Small Coll. 23 (4) (2008) 124–131. URL http://portal.acm.org/citation.cfm?id=1352100.
[50] X. Li, R. Liu, Z. Feng, K. He, Threat modeling-oriented attack path evaluating algorithm, Trans. Tianjin Univ. 15 (3) (2009) 162–167. URL http://www.springerlink.com/content/v76g872558787214/.
[51] P.A. Abdulla, J. Cederberg, L. Kaati, Analyzing the security in the GSM radio network using attack jungles, in: T. Margaria, B. Steffen (Eds.), ISoLA (1), in: LNCS, vol. 6415, Springer, 2010, pp. 60–74.
[52] D. Baca, K. Petersen, Prioritizing countermeasures through the countermeasure method for software security (CM-Sec), in: M.A. Babar, M. Vierimaa, M. Oivo (Eds.), PROFES, in: LNIBP, vol. 6156, Springer, 2010, pp. 176–190.
[53] A. Buldas, P. Laud, J. Priisalu, M. Saarepera, J. Willemson, Rational choice of security measures via multi-parameter attack trees, in: J. López (Ed.), CRITIS, in: LNCS, vol. 4347, Springer, 2006, pp. 235–248.
[54] A. Jürgenson, J. Willemson, Computing exact outcomes of multi-parameter attack trees, in: R. Meersman, Z. Tari (Eds.), OTM Conferences (2), in: LNCS, vol. 5332, Springer, 2008, pp. 1036–1051.
[55] A. Buoni, 2010. Fraud detection: from basic techniques to a multi-agent approach. In: Management and Service Science (MASS), 2010 International Conference on. pp. 1–4.
[56] A. Buoni, M. Fedrizzi, J. Mezei, 2010. A delphi-based approach to fraud detection using attack trees and fuzzy numbers. in: Proceeding of the IASK International Conferences. pp. 21–28.
[57] A. Buoni, M. Fedrizzi, J. Mezei, Combining attack trees and fuzzy numbers in a multi-agent approach to fraud detection, Int. J. Electron. Bus. 9 (3) (2011) 186–202.
[58] T.W. Manikas, M.A. Thornton, D.Y. Feinstein, 2011. Using Multiple-Valued Logic Decision Diagrams to Model System Threat Probabilities. IEEE International Symposium on Multiple-Valued Logic 0, 263–267.
[59] J. Wang, J.N. Whitley, R.C.-W. Phan, D.J. Parish, Unified parametrizable attack tree, Int. J. Inform. Secur. Res. 1 (1) (2011) 20–26.
[60] A. Reinhardt, D. Seither, A. König, R. Steinmetz, M. Hollick, Protecting IEEE 802.11s wireless mesh networks against insider attacks, in: LCN, IEEE, 2012, pp. 224–227.
[61] A. Roy, D.S. Kim, K.S. Trivedi, Attack Countermeasure Trees (ACT): towards unifying the constructs of attack and defense trees, Secur. Commun. Netw. 5 (8) (2012) 929–943.
[62] C. Zhao, Z. Yu, Quantitative analysis of survivability based on intrusion scenarios, in: D. Jin, S. Lin (Eds.), Advances in Electronic Engineering, Communication and Management vol. 2, in: LNEE., vol. 140, Springer, Berlin Heidelberg, 2012, pp. 701–705. http://dx.doi.org/10.1007/978-3-642-27296-7_105.
[63] S. Bortot, M. Fedrizzi, S. Giove, Modelling fraud detection by attack trees and Choquet integral, in: DISA Working Papers 2011/09, Department of Computer and Management Sciences, University of Trento, Italy, 2011, URL http://ideas.repec.org/p/trt/disawp/2011-09.html.
[64] A. Buoni, M. Fedrizzi, Consensual dynamics and choquet integral in an attack tree-based fraud detection system, in: J. Filipe, A.L.N. Fred (Eds.), ICAART (1), SciTePress, 2012, pp. 283–288.
[65] Amenaza, 2001–2013. SecurITree. http://www.amenaza.com/.
[66] Isograph, 2004–2005. AttackTree+. http://www.isograph-software.com/atpover.htm.
[67] P.H. Meland, 2007–2010. SeaMonster. https://sourceforge.net/projects/seamonster/.
[68] N.R. Mead, E.D. Hough, T.R. Stehney II, Security quality requirements engineering (SQUARE) methodology. Tech. Rep. CMU/SEI-2005-TR-009, Carnegie Mellon University, 2005.
[69] Carnegie Mellon University, 2004–2009. SQUARE: System Quality Requirements Engineering. https://www.cert.org/sse/square-tool.html.
[70] E.L. Lazarus, 2010–2011. AttackDog. https://decisionsmith.com/doc/adog.
[71] ACCURATE, 2007. A center for correct usable reliable auditable and transparent elections: annual report 2006. http://accurate-voting.org/wp-content/uploads/2007/02/AR.2007.pdf.
[72] S. Convery, D. Cook, M. Franz, 2004. An attack tree for the border gateway protocol. IETF Internet Draft: http://tools.ietf.org/html/draft-ietf-rpsec-bgpattack-00.
[73] S. Evans, D. Heinbuch, E. Kyule, J. Piorkowski, J. Wallner, Risk-based systems security engineering: stopping attacks with intention, IEEE Secur. Priv. 2 (6) (2004) 59–62.
[74] D.L. Buckshaw, G.S. Parnell, W.L. Unkenholz, D.L. Parks, J.M. Wallner, O.S. Saydjari, Mission oriented risk and design analysis of critical information systems, Milit. Oper. Res. 10 (2) (2005) 19–38.
[75] P.A. Khand, P.H. Seong, 2007. An attack model development process for the cyber security of safety related nuclear digital I&C systems. in: Proceedings of the Korean Nucleary Society (KNS) Fall meeting. Korea.
[76] K. Clark, E. Singleton, S. Tyree, J. Hale, 2008. Strata-Gem: risk assessment through mission modeling. in: Proceedings of the 4th ACM Workshop on Quality of Protection (QoP'08). Alexandria, Virginia, USA, pp. 51–58.

[77] L. Grunske, D. Joyce, Quantitative risk-based security prediction for component-based systems with explicitly modeled attack profiles, J. Syst. Softw. 81 (8) (2008) 1327–1345.
[78] C. Marshall, 2008. Attack trees and their uses in BGP and SMTP analysis. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.122.3609.
[79] Z. Ning, C. Xin-yuan, Z. Yong-fu, X. Si-yuan, Design and application of penetration attack tree model oriented to attack resistance test, in: International Conference on Computer Science and Software Engineering, vol. 3, 2008, pp. 622–626.
[80] G.-Y. Park, C.K. Lee, J.G. Choi, D.H. Choi, Y.J. Lee, K.-C. Kwon, Oct. 2008. Cyber Security Analysis by Attack Trees for a Reactor Protection System. in: Proceedings of the Korean Nuclear Society (KNS) Fall Meeting. Pyeong Chang, Korea.
[81] A.N.P. Morais, E. Martins, A.R. Cavalli, W. Jimenez, Security protocol testing using attack trees, in: CSE (2), IEEE Computer Society, 2009, pp. 690–697.
[82] G. Cagalaban, T. Kim, S. Kim, Improving SCADA control systems security with software vulnerability analysis. in: Proceedings of the 12th WSEAS International Conference on Automatic Control, Modelling & Simulation. ACMOS'10, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2010, pp. 409–414. URL http://dl.acm.org/citation.cfm?id=1844174.1844250.
[83] P. Fernandes, T. Basso, R. Moraes, M. Jino, 2010. Attack trees modeling for security tests in web applications. in: Brazilian Workshop on Systematic and Automated Software Testing. pp. 3–12.
[84] S. McLaughlin, D. Podkuiko, P. McDaniel, Energy theft in the advanced metering infrastructure, in: Proceedings of the 4th International Conference on Critical Information Infrastructures Security. CRITIS'09, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 176–187.
URL http://dl.acm.org/citation.cfm?id=1880551.1880566.
[85] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, P. McDaniel, 2010. Multi-vendor penetration testing in the advanced metering infrastructure. in: Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC'10). Austin, Texas, USA, pp. 107–116.
[86] C.-W. Ten, G. Manimaran, C.-C. Liu, Cybersecurity for critical infrastructures: attack and defense modeling, IEEE Trans. Syst. Man Cybernet. A 40 (4) (2010) 853–865.
[87] A. Morais, A. Cavalli, E. Martins, A model-based attack injection approach for security validation, in: Proceedings of the 4th International Conference on Security of Information and Networks. SIN'11, ACM, New York, NY, USA, 2011, pp. 103–110.
URL http://doi.acm.org/10.1145/2070425.2070443.
[88] M. Sanford, D. Woodraska, D. Xu, Security analysis of FileZilla server using threat models, in: SEKE, Knowledge Systems Institute Graduate School, 2011, pp. 678–682.
[89] M. Warren, S. Leitch, I. Rosewall, 2011. Attack vectors against social networking systems : the Facebook example. in: Proceedings of The 9th Australian Information Security Management Conference. SECAU—Security Research Centre. URL http://hdl.handle.net/10536/DRO/DU:30041837.
[90] H. Suleiman, D. Svetinovic, Evaluating the effectiveness of the security quality requirements engineering (SQUARE) method: a case study using smart grid advanced metering infrastructure, Requirements Eng. (2012) 1–29. http://dx.doi.org/10.1007/s00766-012-0153-4.
[91] EVITA, 2008–2011. E-safety vehicle intrusion protected applications: FP7 project, grant agreement 224275. URL http://www.evita-project.org/.
[92] D.M. Kienzle, Practical computer security analysis (Ph.D. thesis), School of Engineering and Applied Science, University of Virginia, USA, 1998.
[93] D. Pumfrey, The principled design of computer system safety analyses (Ph.D. thesis), Department of Computer Science, University of York, York, UK, 1999, URL http://www.cs.york.ac.uk/~djp/publications/Thesis16.pdf.
[94] F. Moberg, Security analysis of an information system using an attack tree-based methodology (Master's thesis), Chalmers University of Technology, 2000.
[95] N.L. Foster, The application of software and safety engineering techniques to security protocol development (Ph.D. thesis), University of York, 2002.
[96] S.E. Schechter, Computer security strength and risk - a quantitative approach (Ph.D. thesis), Harvard University, Cambridge, Massachusetts, 2004.
[97] A. Opel, Design and implementation of a support tool for attack trees (Master's thesis), Technische Universiteit Eindhoven, Otto-von-Guericke University, Magdeburg, Germany, 2005.
[98] K. Karppinen, Security measurement based on attack trees in a mobile ad hoc network environment (Master's thesis), VTT and University of Oulu, 2005, available at http://www.vtt.fi/inf/pdf/publications/2005/P580.pdf.
[99] K.S. Edge, A Framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees (Ph.D. thesis), Air Force Institute of Technology, Wright Patterson AFB, OH, USA, 2007.
[100] J.H. Espedalen, Attack trees describing security in distributed internet-enabled metrology (Master's thesis), Gjøvik University, 2007.
[101] I. Hogganvik, A graphical approach to security risk analysis (Ph.D. thesis), Faculty of Mathematics and Natural Sciences, University of Oslo, 2007, URL http://heim.ifi.uio.no/~ketils/kst/Theses/2007.Hogganvik.pdf.
[102] T. Mägi, Practical security analysis of e-voting systems (Master's thesis), Tallin University of Technology, Faculty of Information Technology, Department of Informatics, Estonia, 2007, available at http://triinu.net/e-voting/.
[103] P.D. Harrington, Noncooperative potential games to improve network security (Ph.D. thesis). Oklahoma State University, USA, 2010.
[104] A. Jürgenson, Efficient semantics of parallel and serial models of attack trees (Ph.D. thesis), Tallinn University of Technology, Faculty of Information Technology, Department of Informatics, 2010, available at http://digi.lib.ttu.ee/i/?496.
[105] L. Piètre-Cambacédès, Des relations entre sûreté et sécurité (Ph.D. thesis), Télécom ParisTech, 2010.
[106] A. Roy, Attack countermeasure trees: a non-state-space approach towards analyzing security and finding optimal countermeasure sets, (Master's thesis) Duke University, Department of Electrical and Computer Engineering, USA, 2010.
[107] J.R. Nielsen, Evaluating information assurance control effectiveness on an air force supervisory control and data acquisition (SCADA) system (Master's thesis), US Air Force Institute of Technology, 2011, available at http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA541615.
[108] R.T. Ostler, Defensive cyber battle damage assessment through attack methodology modeling (Master's thesis), Air Force Institute of Technology, Department of Electrical and Computer Engineering, USA, 2011.
[109] K.C. Sameer, Attack generation from system models (Master's thesis), Technical University of Denmark, Denmark, 2011.
[110] S.A. Zonouz, Game-theoretic intrusion response and recovery (Ph.D. thesis), University of Illinois at Urbana-Champaign, USA, 2011, available at https://www.ideals.illinois.edu/bitstream/handle/2142/29667/AliariZonouz_Saman.pdf?sequence=1.
[111] A. Buoni, Fraud detection in the banking sector (Ph.D. thesis), Åbo Akademi University, Finland, 2012.

[112] L. Koot, Security of mobile TAN on smartphones (Ph.D. thesis), Radboud University Nijmegen, Faculty of Science, The Netherlands, 2012.
[113] S. Posea, Renewal periods for cryptographic keys (Master's thesis), Eindhoven University of Technology, Department of Mathematics and Computer Science, Eindhoven, The Netherlands, 2012.
[114] Schweitzer Patrick, Attack–defense trees (Ph.D. thesis), University of Luxembourg, 2013.
[115] U.S. Nuclear Regulatory Commission (NRC), 2010. Cyber Security Programs For Nuclear Facilities. Regulatory Guide 5.71.
[116] EUROCAE (European Organisation for Civil Aviation Equipment), 2010. ED-202 – Airworthiness Security Process Specification.
[117] A. van Lamsweerde, 2004. Elaborating security requirements by construction of intentional anti-models. in: 26th International Conference on Software Engineering (ICSE'04). pp. 148–157.
[118] I. Morikawa, Y. Yamaoka, 2011. Threat tree templates to ease difficulties in threat modeling. in: 14th International Conference on Network-Based Information Systems (NBiS'11). pp. 673–678.
[119] T. Sommestad, M. Ekstedt, L. Nordström, Modeling security of power communication systems using defense graphs and influence diagrams, IEEE Trans. Power Deliv. 24 (4) (2009) 1801–1808.
[120] R.J. Anderson, Security Engineering—A Guide to Building Dependable Distributed Systems, first ed., Wiley, 2001.
[121] T.R. Ingoldsby, Understanding risk through attack tree analysis, Comput. Secur. J. 20 (2) (2004) 33–59. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-2542453149&partnerID=40&md5=a06d3ff5d42229c9dd48cdecc74428db.
[122] D.P. Mirembe, M. Muyeba, Threat modeling revisited: improving expressiveness of attack, in: EMS'08: Proceedings of the 2008 Second UKSIM European Symposium on Computer Modeling and Simulation, IEEE Computer Society, Washington, DC, USA, 2008, pp. 93–98.
[123] S. Vidalis, A. Jones, Using vulnerability trees for decision making in threat assessment. Tech. Rep. CS-03-02, School of Computing, University of Glamorgan, Pontypridd, Wales, UK, 2003.
[124] S.C. Patel, J.H. Graham, P.A.S. Ralston, Quantitatively assessing the vulnerability of critical information systems: a new method for evaluating security enhancements, Int. J. Inform. Manag. 28 (6) (2008) 483–491.
[125] I. Ray, N. Poolsapassit, Using attack trees to identify malicious attacks from authorized insiders, in: S. di Vimercati, P. Syverson, D. Gollmann (Eds.), ESORICS'2005, in: LNCS, vol. 3679, Springer, Berlin/Heidelberg, 2005, pp. 231–246. http://dx.doi.org/10.1007/11555827_14.
[126] N. Poolsapassit, I. Ray, Investigating computer attacks using attack trees, in: P. Craiger, S. Shenoi (Eds.), Advances in Digital Forensics III, in: IFIP International Federation for Information Processing, vol. 242, Springer, Boston, 2007, pp. 331–343. http://dx.doi.org/10.1007/978-0-387-73742-3_23.
[127] H. Wang, S. Liu, X. Zhang, An improved model of attack probability prediction system, Wuhan Univ. J. Nat. Sci. 11 (2006) 1498–1502. http://dx.doi.org/10.1007/BF02831806.
[128] J. Wang, R.C.-W. Phan, J.N. Whitley, D.J. Parish, 2010. Augmented attack tree modeling of SQL injection attacks. in: Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME). Chengdu, China, pp. 182–186.
[129] J. Wang, R.C.-W. Phan, J.N. Whitley, D.J. Parish, 2010. Augmented attack tree modeling of distributed denial of services and tree based attack detection method. in: Proceedings of the 10th IEEE International Conference on Computer and Information Technology (CIT 2010). Bradford, UK, pp. 1009–1014.
[130] J. Wang, R.C.-W. Phan, J.N. Whitley, D.J. Parish, London, UK 2010. Quality of detectability (QoD) and QoD-aware AAT-based attack detection. in: Proceedings of the 2010 International Conference for Internet Technology and Secured Transactions (ICITST). Nov., pp. 1–6.
[131] R. Dewri, N. Poolsappasit, I. Ray, D. Whitley, Optimal security hardening using multi-objective optimization on attack tree models of networks, in: Proceedings of the 14th ACM Conference on Computer and Communications Security. CCS'07, ACM, New York, NY, USA, 2007, pp. 204–213. URL http://doi.acm.org/10.1145/1315245.1315272.
[132] R. Dewri, I. Ray, N. Poolsappasit, D. Whitley, Optimal security hardening on attack tree models of networks: a cost-benefit analysis, Int. J. Inf. Secur. 11 (3) (2012) 167–188. http://dx.doi.org/10.1007/s10207-012-0160-y.
[133] A. Jürgenson, J. Willemson, 2007. Processing Multi-Parameter Attacktrees with Estimated Parameter Values. in: [329], pp. 308–319.
[134] J. Willemson, A. Jürgenson, Serial model for attack tree computations, in: D. Lee, S. Hong (Eds.), ICISC, in: LNCS, vol. 5984, Springer, 2010, pp. 118–128. URL http://research.cyber.ee/~jan/publ/serialattack.pdf.
[135] A. Jürgenson, J. Willemson, On fast and approximate attack tree computations, in: Proceedings of the 6th International Conference on Information Security Practice and Experience. ISPEC'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 56–66. http://dx.doi.org/10.1007/978-3-642-12827-1_5.
[136] M. Niitsoo, Optimal adversary behavior for the serial model of financial attack trees, in: Proceedings of the 5th International Conference on Advances in Information and Computer Security. IWSEC'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 354–370. URL http://dl.acm.org/citation.cfm?id=1927197.1927228.
[137] A. Buldas, A. Lenin, New efficient utility upper bounds for the fully adaptive model of attack trees, in: S.K. Das, C. Nita-Rotaru, M. Kantarcioglu (Eds.), GameSec, in: LNCS, vol. 8252, Springer, 2013, pp. 192–205.
[138] A. Andrusenko, 2008. AForest. URL http://research.cyber.ee/~alexander/.
[139] A. Andrusenko, Ründepuude Metoodika Ja Seda Toetav Tarkvaraline Raamistik, Master's thesis, Tallinn University, 2010, available at http://www.cyber.ee/publikatsioonid/20-magistri-ja-doktoritood/loputoeoede-failid/Andrusenko-MA.pdf (in Estonian).
[140] M. Masera, I.N. Fovino, A.D. Cian, Risk, reliability and societal safety, in: T. Aven, J.E. Vinnem (Eds.), Proceedings of the 16th European Safety and Reliability Conference ESREL'07, Taylor & Francis Group, London, 2007, pp. 1–8.
[141] I.N. Fovino, M. Masera, A.D. Cian, Integrating cyber attacks within fault trees, Reliab. Eng. Syst. Safety 94 (9) (2009) 1394–1402. http://dx.doi.org/10.1016/j.ress.2009.02.020.
[142] C. Meadows, 1996. A representation of protocol attacks for risk assessment. in: Proceedings of the DIMACS Workshop on Network Threats. New Brunswick, NJ, USA, pp. 1–10.
[143] H.A. Watson, Launch Control Safety Study. Vol.1, Bell Labs, Murray Hill, NJ, 1961.
[144] M. Stamatelatos, W. Vesely, J. Dugan, J. Fragola, J. Minarick III, J. Railsback, 2002. Fault Tree Handbook with Aerospace Applications. U.S. National Aeronautics and Space Administration (NASA) Handbook: http://www.hq.nasa.gov/office/codeq/doctree/fthb.pdf, version 1.1.
[145] International Electrotechnical Commission (IEC), 2006. Fault tree analysis. IEC 61025, 2nd edn.
[146] C.A. Ericson II, 1999. Fault Tree Analysis—A History. in: Proceedings of the 17th International System Safety Conference (ISSC'99). Orlando, FL, USA.

[147] N.G. Leveson, P.R. Harvey, Software fault tree analysis, J. Syst. Softw. 3 (2) (1983) 173–181. URL http://www.sciencedirect.com/science/article/pii/0164121283900304.
[148] N.G. Leveson, Safeware: System Safety and Computers, Addison-Wesley Professional, 1995, URL http://www.worldcat.org/isbn/0201119722.
[149] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, R. Lutz, A software fault tree approach to requirements analysis of an intrusion detection system, J. Requir. Eng. 7 (4) (2002) 207–220.
[150] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, Y. Wang, X. Wang, N. Stakhanova, Software fault tree and coloured Petri net-based specification, design and implementation of agent-based intrusion detection systems, Int. J. Inform. Comput. Secur. 1 (1/2) (2007) 109–142.
[151] P.J. Brooke, R.F. Paige, Fault trees for security system design and analysis, Computers & Security 22 (3) (2003) 256–264. URL http://www.sciencedirect.com/science/article/pii/S0167404803003134.
[152] Department of Engineering, University of Maryland, ca. 2004. Fault Tree Analysis Programs. http://www.enre.umd.edu/tools/ftap.htm.
[153] J. Pearl, Fusion, propagation, and structuring in belief networks, Artif. Intell. 29 (3) (1986) 241–288. URL http://www.sciencedirect.com/science/article/pii/000437028690072X.
[154] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
[155] R.E. Neapolitan, Learning Bayesian Networks, Prentice Hall, 2003.
[156] F.V. Jensen, T.D. Nielsen, Bayesian Networks and Decision Graphs, second ed., Springer Publishing Company, Incorporated, 2007.
[157] R. Dantu, K. Loper, P. Kolan, 2004. Risk management using behavior based attack graphs. in: International Conference on Information Technology: Coding and Computing (ITCC'04). vol.1. pp. 445–449.
[158] R. Dantu, P. Kolan, Risk management using behavior based Bayesian networks, in: P.B. Kantor, G. Muresan, F. Roberts, D.D. Zeng, F.-Y. Wang, H. Chen, R.C. Merkle (Eds.), ISI, in: LNCS, vol. 3495, Springer, 2005, pp. 115–126.
[159] R. Dantu, P. Kolan, R. Akl, K. Loper, Classification of attributes and behavior in risk management using bayesian networks, IEEE Intell. Secur. Inform. (2007) 71–74.
[160] R. Dantu, P. Kolan, W. ao, J. Cangussu, Network risk management using attacker profiling, Security Commun. Netw. 2 (1) (2009) 83–96. http://dx.doi.org/10.1002/sec.58.
[161] X. An, D. Jutla, N. Cercone, 2006. Privacy intrusion detection using dynamic Bayesian networks. In: Proceedings of the 8th International Conference for Electronic Commerce (ICEC'06). Fredericton, Canada, pp. 208–215.
[162] Q. Althebyan, B. Panda, A knowledge-based Bayesian model for analyzing a system after an insider attack, in: S. Jajodia, P. Samarati, S. Cimato (Eds.), Proceedings of The Ifip Tc 11 23rd International Information Security Conference, in: IFIP International Federation for Information Processing, vol. 278, Springer, Boston, 2008, pp. 557–571. http://dx.doi.org/10.1007/978-0-387-09699-5_36.
[163] P. Xie, J.H. Li, X. Ou, P. Liu, R. Levy, 28 june–july 1 2010. Using Bayesian networks for cyber security analysis. in: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'10). pp. 211–220.
[164] M. Pouly, J. Kohlas, Generic Inference: A Unifying Theory for Automated Reasoning, John Wiley & Sons, Inc, 2011.
[165] S. Arnborg, Efficient algorithms for combinatorial problems on graphs with bounded decomposability—A survey, BIT 25 (1985) 1–23. http://dx.doi.org/10.1007/BF01934985.
[166] H.L. Bodlaender, A linear time algorithm for finding tree-decompositions of small treewidth, in: Proceedings of The Twenty-fifth Annual ACM Symposium on Theory of Computing. STOC'93, ACM, New York, NY, USA, 1993, pp. 226–234. URL http://doi.acm.org/10.1145/167088.167161.
[167] Decision Systems Laboratory, University of Pittsburgh, 1996–2013. GeNIe & SMILE. http://genie.sis.pitt.edu/.
[168] P. Närman, P. Johnson, R. Lagerström, U. Franke, M. Ekstedt, Data collection prioritization for system quality analysis, Electron. Notes Theor. Comput. Sci. 233 (2009) 29–42. URL http://dx.doi.org/10.1016/j.entcs.2009.02.059.
[169] U. Franke, T. Sommestad, M. Ekstedt, P. Johnson, 2008. Defense graphs and enterprise architecture for information assurance analysis. in: Proceedings of the 26th Army Science Conference. Orlando, Florida, USA.
[170] X. Ou, S. Govindavajhala, A.W. Appel, 2005. MulVAL: A logic-based network security analyzer. in: 14th USENIX Security Symposium. pp. 113–128.
[171] M. Scutari, Learning Bayesian Networks with the bnlearn R Package, J. Stat. Softw. 35 (3) (2010) 1–22.
[172] S.H. Houmb, V.N.L. Franqueira, E.A. Engum, Quantifying security risk level from CVSS estimates of frequency and impact, J. Syst. Softw. 83 (9) (2009) 1634–1662. URL http://www.sciencedirect.com/science/article/pii/S0164121209002155.
[173] N. Feng, J. Xie, A Bayesian networks-based security risk analysis model for information systems integrating the observed cases with expert experience, Sci. Res. Essays 7 (10) (2012) 1103–1112.
[174] Y. Liu, H. Man, 2005. Network vulnerability assessment using Bayesian networks. in: Proceedings of SPIE Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005. vol. 5812. Orlando, FL, USA, pp. 61–71.
[175] S. Noel, S. Jajodia, L. Wang, A. Singhal, Measuring security risk of networks using attack graphs, IJNGC 1 (1) (2010) 135–147.
[176] M. Frigault, L. Wang, Measuring network security using Bayesian network-based attack graphs, in: The Proceedings of the 32nd Annual IEEE International Conference on Computer Software and Applications (COMPSAC 08), 2008, pp. 698–703.
[177] M. Frigault, L. Wang, A. Singhal, S. Jajodia, 2008. Measuring network security using dynamic Bayesian network. in: Proceedings of the 4th ACM Workshop on Quality of Protection (QoP'08). Alexandria, Virginia, USA, pp. 23–30.
[178] L. Wang, T. Islam, T. Long, A. Singhal, S. Jajodia, 2008. An attack graph-based probabilistic security metric. in: Proceeedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DAS'2008), LNCS, vol. 5094. London, UK, pp. 283–296.
[179] P. Ammann, D. Wijesekera, S. Kaushik, 2002. Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02). Washington, DC, USA, pp. 217–224.
[180] S. Jajodia, S. Noel, B. O'Berry, Topological analysis of network attack vulnerability, in: Vipin Kumar, Jaideep Srivastava, Aleksandar Lazarevic (Eds.), Managing Cyber Threats: Issues, Approaches, and Challenges, Springer, US, 2005, pp. 247–266.
[181] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, K. Prole, Advances in topological vulnerability analysis, in: Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security. CATCH'09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 124–129.
[182] M.A. McQueen, W.F. Boyer, M.A. Flynn, G.A. Beitel, 2006. Quantitative cyber risk reduction estimation methodology for a small SCADA control system. in: Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS-39). vol.9. Hawaii, USA, pp. 226–237.

[183] M.A. McQueen, W.F. Boyer, M.A. Flynn, G.A. Beitel, 2005. Time-to-compromise model for cyber risk reduction estimation. in: Proceedings of the 1st Workshop on Quality of Protection (QoP'05). Milan, Italy, pp. 49–64.

[184] D.J. Leversage, E.J. Byres, 2008. Estimating a system's mean time-to-compromise. IEEE Security and Privacy 6, 52–60. http://dl.acm.org/citation.cfm?id=1344235.1344300.

[185] D.J. Leversage, E.J. Byres, 2007. Comparing electronic battlefields: using mean time-to-compromise as a comparative security metric. in: Proceedings of the 4th International Conference on Methods, Models, and Architectures for Network Security (MMM-ACNS'07), CCIS 1. St Petersburg, Russia, pp. 213–227.

[186] W. Nzoukou, L. Wang, S. Jajodia, A. Singhal, A Unified framework for measuring a network's mean time-to-compromise, in: SRDS, IEEE, 2013, pp. 215–224.

[187] P. Mell, K. Scarfone, S. Romanosky, Common vulnerability scoring system, IEEE Security & Privacy 4 (6) (2006) 85–89.

[188] S.A. Çamtepe, B. Yener, A formal method for attack modeling and detection. Tech. Rep. TR-06-01, Rensselaer Polytechnic Institute, Troy, NY, USA, 2006.

[189] S.A. Çamtepe, B. Yener, 2007. Modeling and detection of complex attacks. in: Proceedings of the 3rd International Conference on Security and Privacy in Communications Networks (SecureComm 2007). Nice, France, pp. 234–243.

[190] S. Mishra, K. Kant, R.S. Yadav, 2012. Multi tree view of complex attack – stuxnet. in: Proceedings of the ACITY 2012 Conference. Chennai, India, pp. 171–188.

[191] S. Ardi, D. Byers, N. Shahmehri, Towards a structured unified process for software security, in: Proceedings of the 2006 International Workshop on Software Engineering for Secure Systems. SESS'06, ACM, New York, NY, USA, 2006, pp. 3–10. URL http://doi.acm.org/10.1145/1137627.1137630.

[192] D. Byers, S. Ardi, N. Shahmehri, C. Duma, 2006. Modeling software vulnerabilities with vulnerability cause graphs. in: Proceedings of the International Conference on Software Maintenance (ICSM'06). pp. 411–422.

[193] D. Byers, N. Shahmehri, 2007. Design of a Process for Software Security. in: Second International Conference on Availability, Reliability and Security (ARES'07). pp. 301–309.

[194] A. Mammar, A. Cavalli, E. Montes de Oca, S. Ardi, D. Byers, N. Shahmehri, 2009. Modélisation et détection formelles de vulnérabilités logicielles par le test passif. in: 4ème Conférence sur la Sécurité des Architectures Réseaux et des Systèmes d'Information (SAR-SSI). p. 12pp.

[195] N. Chaufette, T. Haag, 2007. Vulnerability cause graphs: a case of study. http://www.ida.liu.se/~TDDD17/oldprojects/2007/projects/3.pdf.

[196] SHIELDS, 2008–2010. GOAT. https://www.ida.liu.se/divisions/adit/security/goat/.

[197] P.A. Khand, 2009. System level security modeling using attack trees. in: Proceedings of the 2nd International Conference on Computer, Control and Communication (IC4). Karachi, Pakistan, pp. 1–6.

[198] J.B. Dugan, S.J. Bavuso, M.A. Boyd, 1990. Fault trees and sequence dependencies. in: Proceedings of the Reliability and Maintainability Annual Symposium (RAMS'90). Los Angeles, CA, USA, pp. 286–293.

[199] J.B. Dugan, S.J. Bavuso, M.A. Boyd, Dynamic fault tree models for fault tolerant computer systems, IEEE Trans. Reliab. 41 (3) (1992) 363–377.

[200] M. Bouissou, 2007. A generalization of dynamic fault trees through Boolean logic driven markov processes (BDMP). In: Proceedings of the 16th European Safety and Reliability Conference (ESREL'07). Stavanger, Norway.

[201] J.B. Dugan, K.J. Sullivan, D. Coppit, Developing a low-cost, high-quality software tool for dynamic fault tree analysis, IEEE Trans. Reliab. 49 (1) (2000) 49–59.

[202] B. Ivanc, T. Klobučar, 2014. Use of the enhanced structural model for attack analysis and education. in: NATO Advanced Research Workshop on "Managing Terrorism Threats to Critical Infrastructure – Challenges for South Eastern Europe".

[203] A. Buldas, R. Stepanenko, Upper bounds for adversaries' utility in attack trees, in: J. Grossklags, J.C. Walrand (Eds.), GameSec, in: LNCS, vol. 7638, Springer, 2012, pp. 98–117.

[204] L. Wen-ping, L. Wei-min, 2011. Space Based information system security risk evaluation based on improved attack trees. in: Third International Conference on Multimedia Information Networking and Security (MINES'11). pp. 480–483.

[205] F. Arnold, H. Hermanns, R. Pulungan, M. Stoelinga, Time-dependent analysis of attacks, in: M. Abadi, S. Kremer (Eds.), POST, in: LNCS, vol. 8414, Springer, 2014, pp. 285–305.

[206] R. Pulungan, H. Hermanns, 2013. APHzip. URL http://depend.cs.uni-saarland.de/tools/aphzip/.

[207] A. van Lamsweerde, S. Brohez, R.D. Landtsheer, D. Janssens, 2003. From system goals to intruder anti-goals: attack generation and resolution for security requirements engineering. in: Proceedings of RHAS'03. pp. 49–56.

[208] A. van Lamsweerde, E. Letier, Handling obstacles in goal-oriented requirements engineering, IEEE Trans. Softw. Eng. 26 (2000) 978–1005. URL http://dl.acm.org/citation.cfm?id=357525.357521.

[209] S. Bistarelli, F. Fioravanti, P. Peretti, Defense trees for economic evaluation of security investments, in: ARES, IEEE Computer Society, 2006, pp. 416–423.

[210] S. Bistarelli, P. Peretti, I. Trubitsyna, Analyzing security scenarios using defence trees and answer set programming, Electron. Notes Theor. Comput. Sci. 197 (2) (2008) 121–129.

[211] K.S. Edge, R.A. Raines, M. Grimaila, R. Baldwin, R. Bennington, C. Reuter, 2007. The Use of Attack and Protection Trees to Analyze Security for an Online Banking System. in: 40th Annual Hawaii International Conference on System Sciences, 2007. (HICSS 2007). p. 144b.

[212] G.C. Dalton II, K.S. Edge, R.F. Mills, R.A. Raines, Analysing security risks in computer and Radio Frequency Identification (RFID) networks using attack and protection trees, Int. J. Security Netw. 5 (2) (2010) 87–95.

[213] R. Cowan, M. Grimaila, R. Patel, 2008. Using Attack and Protection Trees to Evaluate Risk in an Embedded Weapon System. in: Proceedings of the 3rd International Conference on Information Warfare and Security (ICIW 2008). Omaha, Nebraska, USA, pp. 97–108.

[214] G. Ruiz, E. Heymann, E. César, B.P. Miller, Automating threat modeling through the software development lifecycle, 2012.

[215] D. Byers, N. Shahmehri, A cause-based approach to preventing software vulnerabilities, in: Proceedings of the Third International Conference on Availability, Reliability and Security (ARES'08), IEEE Computer Society, Washington, DC, USA, 2008, pp. 276–283.

[216] P.H. Meland, D.G. Spampinato, E. Hagen, E.T. Baadshaug, K.-M. Krister, K.S. Velle, 2008. SeaMonster: Providing tool support for security modeling. in: Norsk Informasjonssikkerhetskonferanse (NISK'08).

[217] A. Roy, D.S. Kim, K.S. Trivedi, 2010 ACT: attack countermeasure trees for information assurance analysis. in: Proceedings of INFOCOM IEEE Conference on Computer Communications Workshops. San Diego, CA, USA, pp. 1–2.

[218] A. Roy, D.S. Kim, K.S. Trivedi, Cyber security analysis using attack countermeasure trees, in: Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research. CSIIRW'10, ACM, New York, NY, USA, 2010, pp. 28:1–28:4. URL http://doi.acm.org.proxy.bnl.lu/10.1145/1852666.1852698.

[219] A. Roy, D.S. Kim, K.S. Trivedi, Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees, in: R.S. Swarz, P. Koopman, M. Cukier (Eds.), DSN, IEEE Computer Society, 2012, pp. 1–12.
[220] K.S. Trivedi, R. Sahner, SHARPE at the age of twenty two, SIGMETRICS Perform. Eval. Rev. 36 (4) (2009) 52–57. URL http://doi.acm.org/10.1145/1530873.1530884.
[221] B. Kordy, S. Mauw, S. Radomirović, P. Schweitzer, in: P. Degano, S. Etalle, J.D. Guttman (Eds.), FAST, in: LNCS, vol. 6561, Springer, 2010, pp. 80–95.
[222] B. Kordy, S. Mauw, S. Radomirović, P. Schweitzer, Attack–defense trees, J. Logic Comput. 24 (1) (2014) 55–87. URL http://logcom.oxfordjournals.org/content/24/1/55.
[223] B. Kordy, M. Pouly, P. Schweitzer, Computational aspects of attack–defense trees, in: Security & Intelligent Information Systems, in: LNCS, vol. 7053, Springer, 2011, pp. 103–116.
[224] B. Kordy, S. Mauw, M. Melissen, P. Schweitzer, Attack–defense trees and two-player binary zero-sum extensive form games are equivalent, in: T. Alpcan, L. Buttyán, J.S. Baras (Eds.), GameSec, in: LNCS, vol. 6442, Springer, 2010, pp. 245–256.
[225] B. Kordy, S. Mauw, P. Schweitzer, Quantitative questions on attack–defense trees, in: ICISC, in: LNCS, vol. 7839, Springer, 2012, pp. 49–64.
[226] B. Kordy, M. Pouly, P. Schweitzer, A probabilistic framework for security scenarios with dependent actions, in: E. Albert, E. Sekerinski (Eds.), iFM, in: LNCS, vol. 8739, Springer International Publishing Switzerland 2014, 2014, pp. 256–271.
[227] P. Kordy, P. Schweitzer, 2012. ADTool. http://satoss.uni.lu/projects/atrees/adtool.
[228] P. Kordy, P. Schweitzer, 2012. The ADTool Manual. http://satoss.uni.lu/software/adtool/manual.pdf.
[229] B. Kordy, P. Kordy, S. Mauw, P. Schweitzer, ADTool: security analysis with attack–defense trees, in: K.R. Joshi, M. Siegle, M. Stoelinga, P.R. D'Argenio (Eds.), QEST, in: LNCS, vol. 8054, Springer, 2013, pp. 173–176.
[230] S. Du, X. Li, J. Du, H. Zhu, 2012. An attack-and-defence game for security assessment in vehicular ad hoc networks. Peer-to-Peer Networking and Applications, 1–14 http://dx.doi.org/10.1007/s12083-012-0127-9.
[231] P. Berander, M. Svahnberg, Evaluating two ways of calculating priorities in requirements hierarchies—an experiment on hierarchical cumulative voting, J. Syst. Softw. 82 (5) (2009) 836–850. http://dx.doi.org/10.1016/j.jss.2008.11.841.
[232] I.S. Moskowitz, M.H. Kang, 1997. An insecurity flow model. in: Proceedings of the 1997 Workshop on New Security Paradigms (NSPW'97). Langdale, Cumbria, UK, pp. 61–74.
[233] International Electrotechnical Commission (IEC), 2006. Analysis techniques for dependability—Reliability block diagram and boolean methods. IEC 61078, 2nd edn.
[234] Y.-S. Wu, B. Foo, B. Matheny, T. Olsen, S. Bagchi, ADEPTS: adaptive intrusion containment and response using attack graphs in an e-commerce environment. Tech. rep, Purdue University, School of Electrical and Computer Engineering, 2003, URL http://www.ece.purdue.edu/~sbagchi/Research/Papers/adepts_dsn04_submit.pdf.
[235] Y.-S. Wu, B. Foo, Y. Mei, S. Bagchi, Collaborative intrusion detection system (cids): a framework for accurate and efficient IDS, in: ACSAC, IEEE Computer Society, 2003, pp. 234–244.
[236] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, E. Spafford, June-1 July 2005. ADEPTS: adaptive intrusion response using attack graphs in an e-commerce environment. in: International Conference on Dependable Systems and Networks (DSN'05). pp. 508–517.
[237] Y.-S. Wu, B. Foo, Y.-C. Mao, S. Bagchi, E. Spafford, Automated aaptive intrusion containment in systems of interacting services. Tech. Rep. Paper 68, Purdue University, School of Electrical and Computer Engineering, West Lafayette, IN 47907-2035, 2005.
[238] T. Sommestad, M. Ekstedt, P. Johnson, 2008. Combining defense graphs and enterprise architecture models for security analysis. in: Proceedings of the 12th IEEE International Conference on Enterprise Distributed Object Computing (EDOC'08). München, Germany, pp. 349–355.
[239] T. Sommestad, M. Ekstedt, P. Johnson, 2009. Cyber Security risks assessment with Bayesian defense graphs and architectural models. in: Proceedings of the 42nd Annual Hawaii International Conference on System Sciences (HICSS-42). Hawaii, USA, pp. 1–10.
[240] M. Ekstedt, T. Sommestad, 2009. Enterprise architecture models for cyber security analysis. in: Proceedings of the IEEE/PES Power System Conference and Exposition (PSCE'09). Seattle, USA, pp. 1–6.
[241] P. Johnson, E. Johansson, T. Sommestad, J. Ullberg, A tool for enterprise architecture analysis, in: EDOC, IEEE Computer Society, 2007, pp. 142–156.
[242] VIKING, 2008–2011. FP7 project, grant agreement 225643. URL http://www.vikingproject.eu.
[243] H. Peine, M. Jawurek, S. Mandel, Security goal indicator trees: a model of software features that supports efficient security inspection, in: HASE'08: Proceedings of the 2008 11th IEEE High Assurance Systems Engineering Symposium, IEEE Computer Society, Washington, DC, USA, 2008, pp. 9–18.
[244] J. Kloos, F. Elberzhager, R. Eschbach, Systematic construction of goal indicator trees for indicator-based dependability inspections, in: 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'10), 2010, pp. 279–282.
[245] S.A. Zonouz, H. Khurana, W.H. Sanders, T.M. Yardley, 2009. RRE: A game-theoretic intrusion Response and Recovery Engine. in: IEEE/IFIP International Conference on Dependable Systems Networks (DSN'09). pp. 439–448.
[246] S.A. Zonouz, A. Sharma, H.V. Ramasamy, Z.T. Kalbarczyk, B. Pfitzmann, K. McAuliffe, R.K. Iyer, W.H. Sanders, E. Cope, 2011. Managing business health in the presence of malicious attacks. in: IEEE/IFIP 41st International Conference on Dependable Systems and Networks Workshops (DSN-W'11). pp. 9–14.
[247] M. Bouissou, J.-L. Bon, A new formalism that combines advantages of fault-trees and Markov models: Boolean logic driven Markov processes, Reliab. Eng. Syst. Safety 82 (2) (2003) 149–163.
[248] L. Piètre-Cambacédès, M. Bouissou, 2010. Beyond attack trees: dynamic security modeling with Boolean logic Driven Markov Processes (BDMP). in: Proceedings of the 8th European Dependable Computing Conference (EDCC-8). Valencia, Spain, pp. 199–208.
[249] L. Piètre-Cambacédès, M. Bouissou, Attack and Defense Modeling with BDMP, in: I. Kotenko, V. Skormin (Eds.), Computer Network Security, in: LNCS, vol. 6258, Springer, 2010, pp. 86–101. URL http://www.springerlink.com/content/47gl0v2158m85340/.
[250] L. Piètre-Cambacédès, M. Bouissou, 2009. The promising potential of the BDMP formalism for security modeling. in: Proceedings of the 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009), Supplemental Volume. Estoril, Portugal, fast Abstract track.
[251] L. Piètre-Cambacédès, Y. Deflesselle, M. Bouissou, 2011. Security modeling with BDMP: from theory to implementation. in: 6th IEEE International Conference on Network and Information Systems Security (SAR-SSI 2011). La Rochelle, France, pp. 1–8.
[252] EDF R & D, 2011–2012. KB3 Platform tools. http://research.edf.com/research-and-the-scientific-community/software/kb3-44337.html.

[253] L. Pièvre-Cambacédès, M. Bouissou, 2010. Modeling safety and security interdepedencies with BDMP (Boolean logic Driven Markov Processes). in: IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010). Istanbul, Turkey, pp. 2852–2861.

[254] C.W. Johnson, Using assurance cases and boolean logic driven markov processes to formalise cyber security concerns for safety-critical interaction with global navigation satellite systems, in: ECEASST 45, 2011, pp. 1–18.

[255] S. Kriaa, M. Bouissou, F. Colin, Y. Halgand, L. Pièvre-Cambacédès, Safety and security interactions modeling using the BDMP formalism: case study of a pipeline, in: A. Bondavalli, F. Di Giandomenico (Eds.), SAFECOMP 2014, in: LNCS, vol. 8666, Springer International Publishing Switzerland 2014, 2014, pp. 326–341.

[256] S. Kriaa, M. Bouissou, L. Pièvre-Cambacédès, 2012. Modeling the stuxnet attack with BDMP: towards more formal risk assessments. in: Martinelli, F., Lanet, J.-L., Fitzgerald, W.M., Foley, S.N. (Eds.), Proceedings of the 7th International Conference on Risks and Security of Internet and Systems (CRiSIS 2012). Cork, Ireland, pp. 1–8.

[257] T. Sommestad, M. Ekstedt, P. Johnson, A probabilistic relational model for security risk analysis, Comput. Secur. 29 (6) (2010) 659–679.

[258] T. Sommestad, M. Ekstedt, H. Holm, The cyber security modeling language: a tool for assessing the vulnerability of enterprise system architectures, IEEE Syst. J. 7 (3) (2013) 363–373.

[259] T. Sommestad, A framework and theory for cyber security assessments (Ph.D. thesis), Industrial Information and Control Systems, QC 20121018, 2012.

[260] N. Friedman, L. Getoor, D. Koller, A. Pfeffer, Learning probabilistic relational models, in: IJCAI, Springer-Verlag, 1999, pp. 1300–1309.

[261] H. Holm, A framework and calculation engine for modeling and predicting the cyber security of enterprise architectures (Ph.D. thesis), Industrial Information and Control Systems, 2014.

[262] F. Johnson, J. Ullberg, M. Buschle, U. Franke, K. Shahzad, P^{2}AMF: predictive, probabilistic architecture modeling framework, in: M. van Sinderen, P.O. Luttighuis, E. Folmer, S. Bosems (Eds.), IWEI, in: LNBIP, vol. 144, Springer, 2013, pp. 104–117.

[263] M. Buschle, J. Ullberg, U. Franke, R. Lagerström, T. Sommestad, A tool for enterprise architecture analysis using the PRM formalism, in: P. Soffer, E. Proper (Eds.), CAiSE Forum, in: LNBIP, vol. 72, Springer, 2011, pp. 108–121.

[264] M. Buschle, P. Johnson, K. Shahzad, The enterprise architecture analysis tool—support for the predictive, probabilistic architecture modeling framework, in: AMCIS, Association for Information Systems, 2013.

[265] D. Byers, N. Shahmehri, Unified modeling of attacks, vulnerabilities and security activities, in: SESS'10: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, ACM, New York, NY, USA, 2010, pp. 36–42.

[266] N. Shahmehri, A. Mammar, E.M. de Oca, D. Byers, A. Cavalli, S. Ardi, W. Jimenez, An advanced approach for modeling and detecting software vulnerabilities, Inf. Softw. Technol. 54 (9) (2012) 997–1013. URL http://www.sciencedirect.com/science/article/pii/S0950584912000535.

[267] xine project, T., 2002–2012. xine multimedia engine. URL http://www.xine-project.org/home.

[268] S. Kumar, E.H. Spafford, 1994. A pattern-matching model for misuse intrusion detection. in: Proceedings of the 17th National Computer Security Conference (NCSC'94). Baltimore, USA, pp. 11–21.

[269] M. Dacier, Vers une évaluation quantitative de la sécurité informatique (Ph.D. thesis), Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS (LAAS), 1994.

[270] J.P. McDermott, 2000. Attack net penetration testing. in: Proceedings of the 2000 Workshop on New Security Paradigms (NSPW'00). Cork, Ireland, pp. 15–21.

[271] V. Horvath, T. Dörges, From security patterns to implementation using petri nets, in: Proceedings of The Fourth International Workshop on Software Engineering for Secure Systems. SESS'08, ACM, New York, NY, USA, 2008, pp. 17–24. URL http://doi.acm.org/10.1145/1370905.1370908.

[272] G.C. Dalton II, R.F. Mills, J.M. Colombi, R.A. Raines, Analyzing attack trees using generalized stochastic petri nets, in: Information Assurance Workshop, 2006, IEEE, West Point, NY, 2006, pp. 116–123.

[273] S. Pudar, G. Manimaran, C.-C. Liu, PENET: a practical method and tool for integrated modeling of security attacks and countermeasures, Comput. Secur. 28 (8) (2010) 754–771.

[274] D. Xu, K.E. Nygard, Threat-driven modeling and verification of secure software using aspect-oriented Petri nets, IEEE Trans. Softw. Eng. 32 (4) (2006) 265–278.

[275] M. Dacier, Y. Deswarte, Privilege graph: an extension to the typed access matrix model, in: D. Gollmann (Ed.), ESORICS'1994, in: LNCS, vol. 875, Springer, 1994, pp. 319–334. URL http://dx.doi.org/10.1007/3-540-58618-0_72.

[276] M. Dacier, Y. Deswarte, M. Kaâniche, Models and tools for quantitative assessment of operational security, in: S.K. Katsikas, D. Gritzalis (Eds.), SEC, in: IFIP Conference Proceedings, vol. 54, Chapman & Hall, 1996, pp. 177–186.

[277] A.N. Zakrzewska, E.M. Ferragut, 2011. Modeling cyber conflicts using an extended Petri Net formalism. in: Computational Intelligence in Cyber Security (CICS), 2011 IEEE Symposium on. pp. 60–67.

[278] C. Phillips, L.P. Swiler, 1998. A graph-based system for network-vulnerability analysis. in: Proceedings of the 1998 Workshop on New Security Paradigms (NSPW'98). Charlottesville, Virginia, USA, pp. 71–79.

[279] L.P. Swiler, C. Phillips, D. Ellis, S. Chakerian, 2001. Computer-attack graph generation tool. DARPA Information Survivability Conference and Exposition II (DISCEX'01) 2, 307–321.

[280] O. Sheyner, J. Haines, S. Jha, R. Lippmann, J.M. Wing, 2002. Automated generation and analysis of attack graphs. in: Proceedings of the IEEE Symposium on Security and Privacy (S&P' 02). Oakland, California, USA, pp. 273–284.

[281] O. Sheyner, Scenario graphs and attack graphs (Ph.D. thesis), Carnegie Mellon University (CMU), Pittsburgh, PA, 2004.

[282] S. Noel, S. Jajodia, B. O'Berry, M. Jacobs, 2003. Efficient minimum-cost network hardening via exploit dependency graphs. in: Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03). Las Vegas, NV, USA, pp. 86–95.

[283] R. Lippmann, K.W. Ingols, 2005. An annotated review of past papers on attack graphs. Project Report ESC-TR-2005-054, Massachusetts Institute of Technology (MIT), Lincoln Laboratory.

[284] L. Wang, C. Yao, A. Singhal, S. Jajodia, Interactive analysis of attack graphs using relational queries, in: E. Damiani, P. Liu (Eds.), Data and Applications Security XX, in: LNCS, vol. 4127, Springer, Berlin Heidelberg, 2006, pp. 119–132. URL http://dx.doi.org/10.1007/11805588_9.

[285] X. Ou, W.F. Boyer, M.A. McQueen, 2006. A scalable approach to attack graph generation. in: Proceedings of the 13th ACM conference on Computer and Communications Security (CCS'06). Alexandria, Virginia, USA, pp. 336–345.

[286] K.W. Ingols, R. Lippmann, K. Piwowarski, 2006. Practical Attack graph generation for network defense. in: Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC'06). Washington, DC, USA, pp. 121–130.

[287] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, J. Wing, 2006. Ranking attack graphs. in: Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID'06), LNCS, vol. 4219. Hamburg, Germany, pp. 127–144.

[288] S. Malhotra, S. Bhattacharya, S.K. Ghosh, 2008. A vulnerability and exploit independent approach for attack path prediction. in: Proceedings of the IEEE 8th International Conference on Computer and Information Technology Workshops. Sydney, Australia, pp. 282–287.

[289] P.K. Manadhata, An attack surface metric (Ph.D. thesis), Carnegie Mellon University, 2008.

[290] S. Noel, S. Jajodia, Managing attack graph complexity through visual hierarchical aggregation, in: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security (VizSEC'04), George Mason University, Fairfax, VA, USA, 2004, pp. 109–118.

[291] S. Noel, M. Jacobs, P. Kalapa, S. Jajodia, 2005. Multiple coordinated views for network attack graphs. in: Proceedings of the 2005 IEEE Workshop on Visualization for Computer Security (VizSEC 05). Minneapolis, USA, pp. 99–106.

[292] L. Williams, R. Lippmann, K.W. Ingols, 2007. An interactive attack graph cascade and reachability display. in: Proceedings of the 2007 Workshop on Visualization for Computer Security (VizSEC'07). Sacramento, CA, USA, pp. 221–236.

[293] J. Homer, A. Varikuti, X. Ou, M.A. McQueen, 2008. Improving attack graph visualization through data reduction and attack grouping. in: Proceedings of the 5th International Workshop on Visualization For Computer Security (VizSEC'08). Cambridge, MA, USA, pp. 68–79.

[294] I. Kotenko, M. Stepashkin, Analyzing network security using malefactor action graphs, Int. J. Comput. Sci. Netw. Secur. 6 (6) (2006) 226–235.

[295] S. Braynov, M. Jadliwala, 2003. Representation and analysis of coordinated attacks. in: Proceedings of the 2003 ACM Workshop on Formal Methods in Security Engineering (FMSE'03). Washington, D.C., USA, pp. 43–51.

[296] L. Wang, S. Noel, S. Jajodia, Minimum-cost network hardening using attack graphs, Comput. Commun. 29 (18) (2006) 3812–3824.
URL http://dx.doi.org/10.1016/j.comcom.2006.06.018.

[297] L. Wang, A. Singhal, S. Jajodia, Measuring the overall security of network configurations using attack graphs, in: S. Barker, G.-J. Ahn (Eds.), Data and Applications Security XXI, in: LNCS., vol. 4602, Springer, Berlin/Heidelberg, 2007, pp. 98–112. http://dx.doi.org/10.1007/978-3-540-73538-0_9.

[298] L. Wang, A. Singhal, S. Jajodia, Toward measuring network security using attack graphs, in: Proceedings of the 2007 ACM workshop on Quality of Protection. QoP '07, ACM, New York, NY, USA, 2007, pp. 49–54.
URL http://doi.acm.org/10.1145/1314257.1314273.

[299] G.R. Louthan IV, Hybrid attack graphs for modeling cyber-physical systems (Master's thesis), University of Tulsa, USA, 2011.

[300] J. Dawkins, J. Hale, 2004. A systematic approach to multi-stage network attack analysis. in: Proceedings of the 2nd IEEE International Information Assurance Workshop (IAWA'04). Charlotte, NC, USA, pp. 48–56.

[301] K. Clark, S. Tyree, J. Dawkins, J. Hale, 2004. Qualitative and quantitative analytical techniques for network security assessment. in: Proceedings of the 5th IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW'04). West Point, USA, pp. 321–328.

[302] L. Samarji, F. Cuppens, N. Cuppens-Boulahia, W. Kanoun, S. Dubus, in: G. Wang, I. Ray, D. Feng, M. Rajarajan (Eds.), CSS, in: LNCS, vol. 8300, Springer, 2013, pp. 132–150.

[303] J.A. Pinto, Temporal reasoning in the situation calculus (Ph.D. thesis), University of Toronto, Ontario, Canada, 1994, AAINN92616.

[304] J.P. McDermott, C. Fox, 1999. Using abuse case models for security requirements analysis. in: Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99). Phoenix, USA, pp. 55–64.

[305] G. Sindre, A.L. Opdahl, 2000. Eliciting Security requirements by misuse cases. in: Proceedings of 37th International Conference on Technology of Object-Oriented Languages and Systems (TOOLS-PACIFIC 2000). Sydney, Australia, pp. 120–131.

[306] G. Sindre, A.L. Opdahl, 2001. Templates for misuse case description. in: Proceedings of the 7th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2001). Interlaken, Switzerland, pp. 125–136.

[307] G. Sindre, A.L. Opdahl, G.F. Brevik, 2002. Generalization/specialization as a structuring mechanism for misuse cases. in: Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS'02). Raleigh, NC, USA.

[308] I. Alexander, Misuse cases: use cases with hostile intent, IEEE softw. 20 (1) (2003) 58–66.

[309] G. Sindre, A.L. Opdahl, Eliciting security requirements with misuse cases, J. Requirements Engineering 10 (2005) 34–44. http://dx.doi.org/10.1007/s00766-004-0194-4.

[310] L. Røstad, 2006. An extended misuse case notation: Including vulnerabilities and the insider threat. in: Proceedings of the 12th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2006). Luxembourg, Grand-Duchy of Luxembourg, pp. 33–43.

[311] D.J. Firesmith, Security Use Cases, J. Object Technol. 2 (3) (2003) 53–64. URL
http://www.jot.fm/issues/issue_2003_05/column6.

[312] CC, 2012. Common Criteria for Information Technology Security Evaluation (version 3.1, revision 4). ISO/IEC 15408. URL http://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf.

[313] M.H. Diallo, J. Romero-Mariona, S.E. Sim, T.A. Alspaugh, D.J. Richardson, 2006. A comparative evaluation of three approaches to specifying security requirements. in: Proceedings of the 12th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2006). Luxembourg, Grand-Duchy of Luxembourg.

[314] A.L. Opdahl, G. Sindre, Experimental comparison of attack trees and misuse cases for security threat identification, Inform. Softw. Technol. 51 (5) (2009) 916–932.

[315] I.A. Tøndel, J. Jensen, L. Røstad, 2010. Combining misuse cases with attack trees and security activity models. in: International Conference on Availability, Reliability and Security. IEEE Computer Society, Los Alamitos, CA, USA, pp. 438–445.

[316] P.H. Meland, I.A. Tøndel, J. Jensen, 2010. Idea: reusability of threat models—two approaches with an experimental evaluation. in: International Symposium on Engineering Secure Software and Systems (ESSoS). Pisa, Italy, pp. 114–122.

[317] P. Kárpáti, G. Sindre, A.L. Opdahl, 2010. Visualizing cyber attacks with misuse case maps. in: Proceedings of the 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2010). Essen, Germany, pp. 262–275.

[318] V. Katta, P. Kárpáti, A.L. Opdahl, C. Raspotnig, G. Sindre, Comparing two techniques for intrusion visualization, in: P. van Bommel, S. Hoppenbrouwers, S. Overbeek, E. Proper, J. Barjis (Eds.), PoEM, in: Lecture Notes in Business Information Processing, vol. 68, Springer, 2010, pp. 1–15.

[319] Kárpáti Péter, Sindre Guttorm, Andreas L. Opdahl, Towards a hacker attack representation method, in: Proceedings of the 5th ICSOFT Conference, 2010, pp. 92–101.

[320] G. Sindre, 2007. Mal-activity diagrams for capturing attacks on business processes. in: Proceedings of the 13th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2007), LNCS, vol. 4542. Trondheim, Norway, pp. 355–366.

[321] P. Kárpáti, G. Sindre, R. Matulevicius, Comparing misuse case and mal-activity diagrams for modelling social engineering attacks, IJSSE 3 (2) (2012) 54–73.

[322] K. Daley, R. Larson, J. Dawkins, 2002. A Structural Framework for Modeling Multi-Stage Network Attacks. in: ICPP Workshops. IEEE Computer Society, pp. 5–10.

[323] P. Johnson, R. Lagerström, P. Närman, M. Simonsson, Enterprise architecture analysis with extended influence diagrams, Inform. Syst. Front. 9 (2–3) (2007) 163–180.

[324] J.E. Matheson, R.A. Howard, An Introduction to Decision Analysis, Strategic Decisions Group, Menlo Park, CA, 1968.

[325] B.C. Ezell, S.P. Bennett, D. von Winterfeldt, J. Sokolowski, A.J. Collins, Probabilistic risk analysis and terrorism risk, Risk analysis an official publication of the Society for Risk Analysis 30 (4) (2010) 575–589. http://www.ncbi.nlm.nih.gov/pubmed/20522198.

[326] R. Lagerström, P. Johnson, P. Närman, Extended Influence Diagram Generation, in: R. Jardim-Gonçalves, J.P. Müller, K. Mertins, M. Zelm (Eds.), IESA, Springer, 2007, pp. 599–602.

[327] SHIELDS, 2010. Final SHIELDS approach guide—Deliverable D1.4. URL http://www.shields-project.eu/files/docs/D1.4%20Final%20SHIELDS%20Approach%20Guide.pdf.

[328] GraMSec, 2014. The First International Workshop on Graphical Models for Security. URL http://gramsec.uni.lu/.

[329] A. Miyaji, H. Kikuchi, K. Rannenberg (Eds.), Advances in Information and Computer Security, Second International Workshop on Security, IWSEC 2007, Nara, Japan, October 29–31, 2007, Proceedings, in: LNCS, vol. 4752, Springer, 2007.